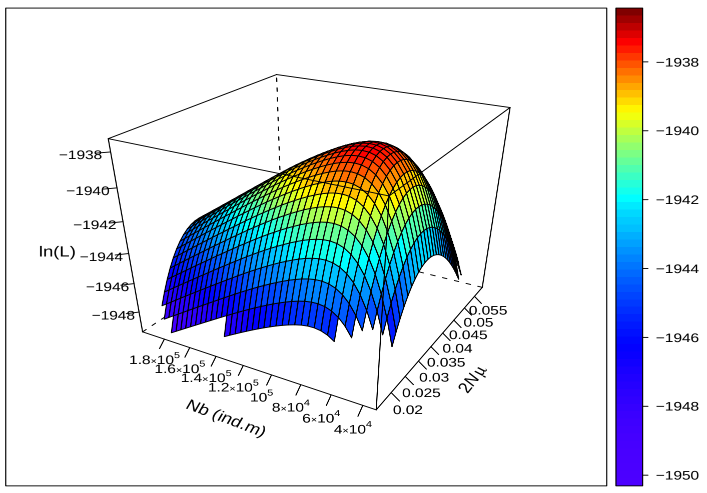
Migraine version 0.6

for Linux/Windows/MacOs

Short documentation

October 1, 2020



Introduction

The `Migraine` program allows likelihood analyses of genetic data, with a focus on the inference of dispersal for spatially structured populations and historical events for isolated panmictic populations. It is mainly designed for allelic type data sets, but short non-recombining DNA sequence data analyses can be also analyzed under some demographic models. Moreover, analyses combining different type of markers, e.g. microsatellites and DNA sequences, are also allowed. The demographic models currently implemented in this program are (1) simple models of isolation by distance (`IBD`) in linear and two-dimensional habitats, as described in (Rousset & Leblois, 2007, 2012), which includes the finite island and stepping stone models as a special cases; (2) a single population model (`OnePop`); (3) a single population model with a single continuous past variation in population size (`OnePopVarSize`), as described in (Leblois *et al.*, 2014). This model can be used to infer parameters under scenarios of past contraction or expansion in population size; (3) a single population model with two past variation in population size (the first is discrete followed forward in time by a continuous one) (`OnePopFounderFlush`), as described in (Rousset *et al.*, 2018). This model can typically be used for scenarios with a founder event followed by an expansion (that gave its name to the model), often observed in invasion or epidemiolocal processes, or for any other combination with two past changes in size; (5) an n-population model with a constant (in time) migration matrix is also implemented (`Npop`) but has only been tested in its simplest configuration with two populations as described in de Iorio *et al.* (2005) (`2pop`). Currently, a K-alleles mutation model is implemented for all demographic models and two stepwise mutation models (`SMM` and `GSM`) are implemented to some extent for models with one or two populations (i.e. `OnePop`, `OnePopVarSize`, `OnePopFounderFlush`, for both models, and `2pop` for the `SMM` only).

To estimate model parameters, `Migraine` infers likelihood. Point estimates, confidence intervals and likelihood ratio tests are then computed from the likelihood surface. A practical problem for applying these well-known methods to population genetic inferences is that the likelihood itself has to be estimated by simulation. In `Migraine` this is accomplished by the class of importance sampling algorithms defined by de Iorio & Griffiths (2004a,b) and de Iorio *et al.* (2005). Alternatively, approximations known as PAC-likelihood (Product of Approximate Conditional likelihoods, Li & Stephens, 2003) can be used. Cornuet & Beaumont's (2007) version of PAC-likelihood, which is based on quantities inherent to the importance sampling algorithms, is implemented in `Migraine`. Finally, we also implemented a resampling procedure in the importance sampling algorithm, based on the work of Liu *et al.*

(2001) and Liu (2008). For the moment, this resampling procedure is only implemented for the OnePopVarSize model under a SMM mutation model in Migraine. It is described in detail and tested in Merle *et al.* (2017).

Migraine is designed to interact with the R software for data analysis (R Core Team, 2013). R is free software available for all major operating systems. This documentation assumes no previous knowledge of R. All analyses can be performed, and decent graphics can be produced, without any knowledge of the R language. However, to see how this can be done, it is essential to install R and to perform the session examples.

There are two versions of this documentation. The short version first describes the installation steps and two worked examples (under the LinearIBD and OnePopVarSize models), for a quick start. This is followed by a minimal description of the methods used; a description of the statistical models implemented and of their specificities (e.g., the neighborhood parameter for isolation by distance models), followed by a summary of the canonical parameter order for each model; a similar description of data input; and a systematic description of the most important settings. Finally, additional worked examples are shown. The long version provides additional information on all the above topics, including some troubleshooting advice, and instructions for running multiple Migraine processes. You are reading the *short* version.

Here is the more formal Table of contents:

1 Quick start

1.1 Requirements

Migraine should run on most reasonably recent operating systems with a C++ compiler and a R software installed.

`Migraine` has limited memory needs. However, memory issues can become a problem when running the `R` code, and access to 64-bit processors with large amounts of RAM may be handy in that case.

If you plan to run several concurrent `Migraine` processes in the same directory, then read section 8 of the long version of this documentation.

1.2 Installation

1.2.1 `Migraine`

Windows users can run the executable `Migraine.exe`.

For Linux users, compile the sources by either

```
g++  -O3 -o migraine *.cpp
```

or by

```
g++ -DNO_MODULES -o migraine latin.cpp -O3
```

(the second compilation command will generate a smaller executable file). This should work on most Unix-based systems, including Mac OS X. If you use the `clang` compiler, then you may need the `-std` option as in

```
g++ -DNO_MODULES -std=c++11 -o migraine latin.cpp -O3
```

1.2.2 The `R` statistical software

A recent version of `R` must be installed, including some packages available from the CRAN websites, in particular the `blackbox` package. This is quite straightforward if you are familiar with `R` installation issues. If not, the following may help you.

All `R` sources and documentation can be found on the CRAN website. `blackbox` is a standard R package available on CRAN, so to install it something as simple as

```
install.packages("blackbox")
```

may suffice. For **Windows**, this will install precompiled binaries for the package, and other required R packages will be automatically installed. Under **linux**, you may need to help yourself a bit more. In particular, installation of the `rcdd` package requires the `gmplib` (GNU Multiple Precision) library. If it is not already installed, try something like `apt-get install libgmp3-dev` if you have set an adequate repository, or else follow the instructions on www.gmplib.org.

The run time of the `R` code may be **substantially** reduced if you compile the sources with the the `-O3` compiler option for compiler optimisation in `g++` or `clang`. Unfortunately, this is not the default setting in most `R` installation we have used. You can change this locally by creating a file containing the line `CXXFLAGS= -O3` which must be appropriately named and put in the appropriate directory (this is in principle explained in the `R admin` documentation). For Windows (64 bit version) one can put them in the `Makevars.site` file in the `\etc\x64` subdirectory of the current `R` installation. Under Linux, this should be in `~/.R/Makevars`. Then you need to install the package from source, not from binaries, by using

```
install.packages("blackbox",type="source")
```

On Windows, installing this from source means that (1) you may need to read the documentation for `install.packages` (particularly its `dependencies` argument); (2) you **need** to have installed the Rtools **first**, which in now pretty straightforward (Download and run the installer from **here** and follow instructions). Then you can compile the `blackbox` library as any other package from CRAN: launch `R`, and run the above command line. Do not use the `R` GUI menu command to install the package. Check that the installation succeeds (it should terminate with the message `DONE (blackbox)`. If it fails, check that you have correctly installed the Rtools by trying to install another package that requires compilation, e.g.

```
install.packages("lpSolveAPI",type="source")
```

1.3 Using `Migraine`

We first present two minimal worked examples of inference, one for isolation by distance, one for a single past change in population size of a single population (i.e., `OnePopVarSize` model).

1.3.1 Minimal example for isolation by distance

In this example, `Migraine` will analyze the data from a damselfly population (Watts *et al.*, 2007). Likelihoods will be computed for the three parameters of a simple model of linear isolation by distance.

Copy the provided `migraine.txt` and the sample file `IVCP` (that can be found in the folder `firstSession/IBD_IVCP/`) into an empty folder. Make the `migraine` executable accessible from this folder by whatever mean suitable for your operating system. Launch the executable (simply by entering its name on the command line). Wait for completion of the computation. The likelihood computation should last only a few seconds. The most important files it generates are `pointls_1.txt` and `migraine_1.R`.

The `R` analysis will take a few minutes, unless it fails if `R` and its additional packages were not properly installed. In the latter case, see the long version of this documentation for further advice

If everything goes well, several output files will be produced. The main results are saved in the `results_1.txt` file, which looks like:

```
----------------------------------------------------------------
Migraine 0.5 (Built on Sep  2 2016 at 11:16:56)
blackbox, version 1.0.12 loaded
R code run on  Thu Sep 08 19:01:52 2016

Data file: IVCP
Settings file: migraine.txt

Geographic bin width= 692.006
Demographic model: IBD 1D
Canonical parameters: 2Nmu 2Nm g
* N stands for number of gene copies,
    i.e. 2N = 4 x [the number of diploid individuals] *

(!) Few points in upper 11.91 [ln(L) units] range:
   only 320 points in this range.
(!) Only 15 points have a predicted likelihood
     in the upper 1.921 [ln(L) units] range.
    (this threshold corresponds to the 0.05 chi-square threshold with 1 df);
    It is advised to compute more points in order to obtain good CIs.

    Some high profile likelihoods are extrapolated in the 2Nmu, Nb profile

*** Confidence intervals ***

95%-coverage confidence interval for 2Nmu : [ 0.363 -- 0.643 ]
95%-coverage confidence interval for 2Nm : [ 45.47 -- 123.4 ]
95%-coverage confidence interval for g : [ 0.301 -- 0.997 ]
95%-coverage confidence interval for Nb : [ 167792 -- 8087569688 ]

*** Point estimates ***

   2Nmu    2Nm      g
  0.481  68.96  0.802

      Neighborhood: 2190463 ind.m
```

```
Normal ending.
-----------------------------------------------------------------
```

1.3.2 Minimal example for the `OnePopVarSize` model

In this second example, `Migraine` will analyze the data from a Soay sheep population, kindly provided by J. Pemberton and analyzed with `Migraine` in Rousset *et al.* (2018). Likelihoods will be computed for the three parameters of the `OnePopVarSize` model: a panmictic isolated population with a single past change in size, for which we fixed the `pGSM` parameter value to allow faster analyses with only three parameters.

Copy the provided `migraine.txt` and the sample file `Soay.txt` (that can be found in the folder `firstSession/OPVS_Soay/`) into an empty folder. Make the `migraine` executable accessible from this folder by whatever mean suitable for your operating system. Launch the executable (simply by entering its name on the command line). Wait for completion of the computation. The likelihood computation should last a few minuts. The most important files it generates are `pointls_1.txt` and `migraine_1.R`.

Then the `R` analysis, which should automatically be launched by the `Migraine` executable, will also take a few minutes, unless it fails if `R` and its additional packages were not properly installed. In the latter case, see the long version of this documentation for further advice

If everything goes well, several output files will be produced by the `R` analysis. The main results are saved in the `results_1.txt` file, which looks like:

```
-----------------------------------------------------------------
Migraine 0.5 (Built on Feb 21 2017 at 18:05:04)
blackbox, version 1.0.18 loaded
R code run on  Thu Mar 30 15:13:15 2017

Data file: Soay.txt
Settings file: migraine.txt

Demographic model: OnePopVarSize
Canonical parameters: pGSM 2Nmu Tg/2N Dg/2N 2Nancmu
* N stands for number of gene copies,
    i.e. 2N = 4 x [the number of diploid individuals] *

(!) Few points in upper 33.02 [ln(L) units] range:
   only 387 points in this range.
(!) Only 57 points have a predicted likelihood
```

```
     in the upper 1.921 [ln(L) units] range.
    (this threshold corresponds to the 0.05 chi-square threshold with 1 df);
    It is advised to compute more points in order to obtain good CIs.

*** Confidence intervals ***

95%-coverage confidence interval for 2Nmu : [ 0.158 -- 0.551 ]
95%-coverage confidence interval for Dg/2N : [ 0.248 -- 0.94 ]
95%-coverage confidence interval for 2Nancmu : [ 3.424 -- 15.86 ]
95%-coverage confidence interval for Nratio : [ 0.0206 -- 0.0956 ]
95%-coverage confidence interval for Dg*mu : [ 0.0475 -- 0.444 ]

*** Point estimates ***

     pGSM     2Nmu    Tg/2N    Dg/2N  2Nancmu
      0.5    0.327        0     0.56    7.465

      N ratio: 0.0438

      Dg*mu: 0.183

Normal ending.________________________________________________________________
```

1.3.3 Going further into the results of those minimal worked examples

For both minimal examples described above, further information is reported in several ways (detailed later). When using the R GUI, beware that several graphic windows will be produced on top of each other: you need to move each window to see the previous one.

`R` should produce several plots in the `Rplots_1.eps` file, some of which are shown in Fig. 1 and 2. The following types of plots are produced:

- Raw one-dimensional projections of the cloud of points for each parameter (first plot in Fig. 1 and 2). These plots are not very important unless something goes wrong.[1] Nevertheless, they allow a quick examination of the results, in comparison to the next plots which are slower to produce;
- contour plots of the likelihood surface, where one parameter estimate

[1]These diagnostic plots are a bit messy, as two different scales may be shown on the same frame. The traditional lower/left scales spans all points, shown in grey, the upper/right scale spans points selected for kriging, shown in black; points selected for generalized cross-validation are circled in red.

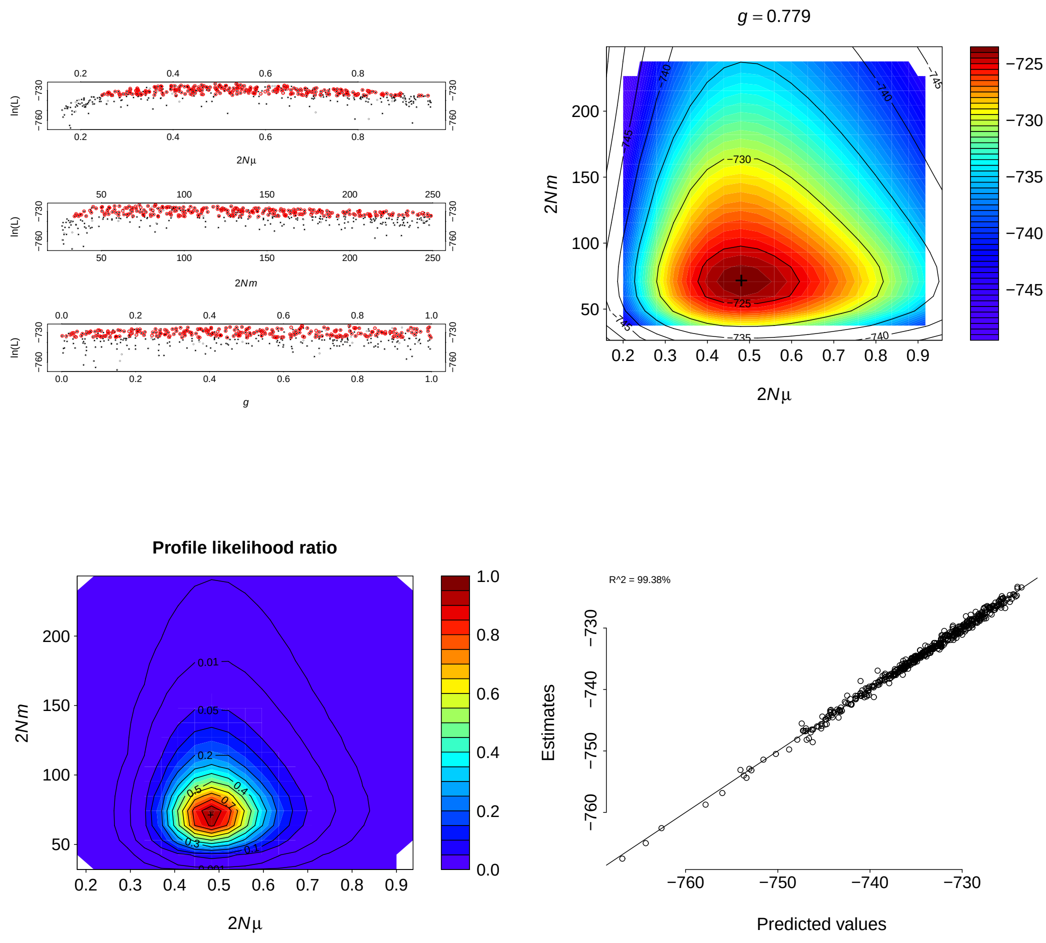


Figure 1: Four types of plots produced by `Migraine` under the `linearIBD` model.
These are parts of the graphic output from analysis of example file `IVCP` as described in the text.

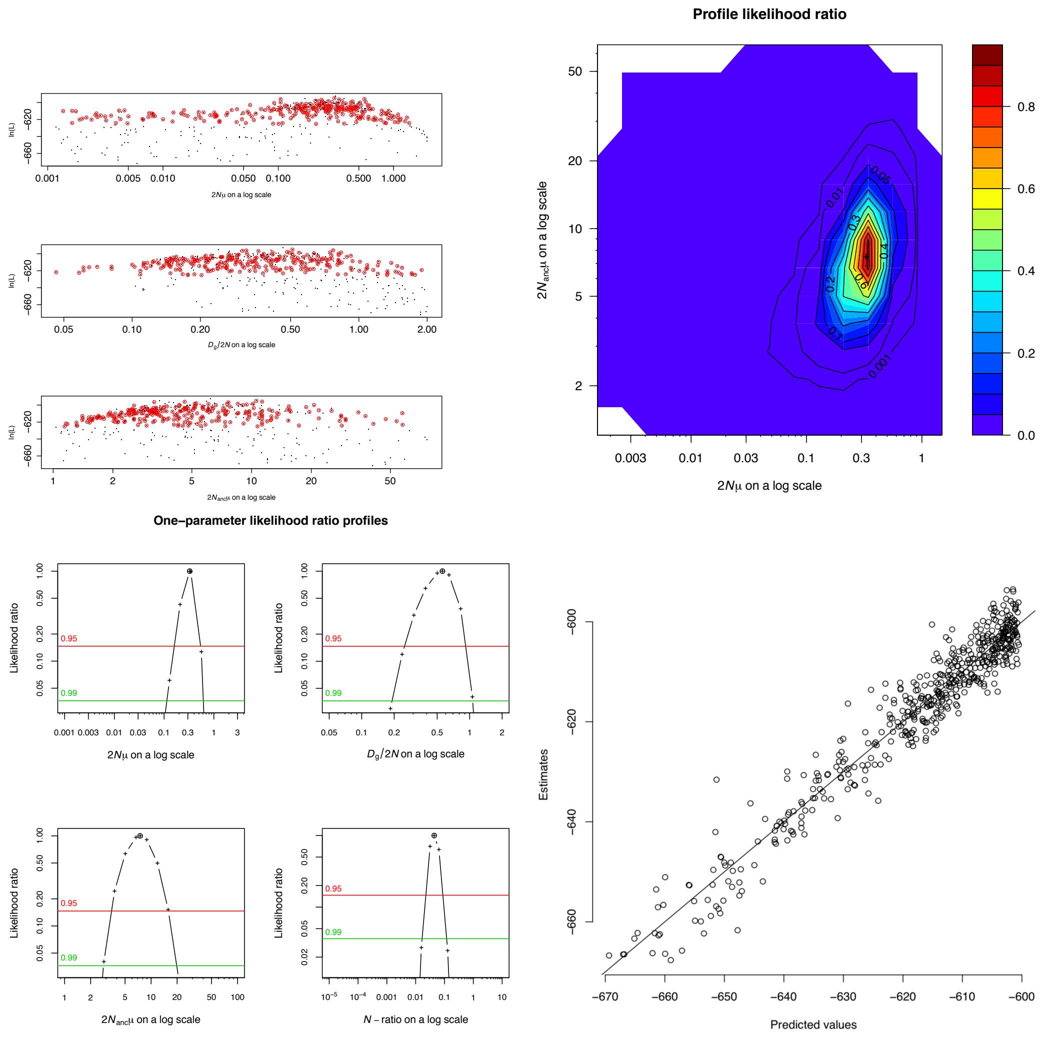


Figure 2: Four types of plots produced by `Migraine` under the `OnePopVarSize` model.
These are parts of the graphic output from analysis of the minimal Soay sheep example, after the second iteration (see Section 1.5), as described in the text.

has been fixed to its maximum likelihood value (hence, slice in the three-dimensional parameter space; second plot in Fig. 1). These can also be shown as perspective (or "3D") plots, although the latter are better suited to make a showy image (see the cover page of this documentation) than to carry a clear message;

- 2D profile likelihood regions for pairs of parameters (third plot in Fig. 1 and second plot in Fig. 2);

- 1D likelihood profiles for each canonical parameter and for some composite parameters of the model (third plot in Fig. 2 for the `OnepopVarSize` example; not produced in the first `LinearIBD` example). These may be produced at two steps: before and after the computation of the 2D profiles. 1D profiles computed after 2D profiles take advantage of the computation of the latter to circumvent problems with local maxima in maximization steps. Therefore, these 1D profiles are more reliable and should be retained.

- an "observed vs. predicted" diagnostic plot which should look like an ideal regression line with 1:1 slope, and Gaussian-distributed error (fourth plot in Fig. 1 and 2). As explained later, the likelihood surface is inferred by a smoothing operation on the likelihood points first computed by `Migraine`. In general the surface should not pass through the points and this plots show the difference.

These two examples only serve as a quick introduction to `Migraine`, and some the results may be far from perfect. See Section 6.3 for more explanation of the graphics, and Sections 4 and 8 for more examples and hints for good results for the different demographic models.

1.3.4 The settings file and the command line

At this point, it is worth having a look at the `migraine.txt` file. For the first example under the `LinearIBD` model, it should look like:

```
DemographicModel=LinearIBD
statistic=PAC
PointNumber=512
Nrunsperpoint=5
GeoUnit= ind.m
GenepopFileName=IVCP
GeoDistanceBins=5
onedimCI=twoNmu,twoNm,g,Nb
```

```
writeSequence=Over
LowerBound=0.16,25,0.
Upperbound=0.96,250,0.999
### for the second iteration:
#WriteSequence=ReadPoints,Append
```

and for the second example under the `OnepopVarSize` model, it should look like:

```
DemographicModel=OnePopVarSize
Statistic=PACanc
WriteSequence=Over
PointNumber=300
NRunsPerPoint=20
GenepopFileName=Soay.txt
MutationalModel=GSM
GivenK=50
StepSizes=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
GridSteps=15
Plots=Allprofiles
1DCI=2Nmu,Dchange,2Nancmu,Nratio,Dgmu
LowerBound=0.5,0.001,0.0,0.1,1
UpperBound=0.5,2.0,0.0,2.0,80.0
SamplingScale=,logscale,,logscale,logscale
## For the second iteration, uncomment next line
#writesequence=ReadPoints,Append
```

The significance of these settings, and many more, will be explained in the next sections.

However, we should insist on **two very important points for analysis under time-inhomogeneous models** such as the `OnePopVarSize` model : (1) the user should be cautious about any **population structure** detected in his data set because inferences under those time-in-homogeneous models are very sensitive to population structure. Only the absence of any population structure allows to pool samples from different geographical places (Leblois *et al.*, 2014); (2) inferences under those models are also very **sensitive to mutational processes** (see Leblois *et al.*, 2014). The `StepSizes` keyword in the above settings corresponds to the size of the microsatellite motive for each locus (i.e. 2 for di-nucleotides, 3 for tri-nucleotides, 4 for tetra-nuceoltides, etc). If this `StepSizes` setting is absent, `Migraine` will compute the smallest compatible size for each locus but may fail (i.e. found a motive size of 1) if some mutations do not strictly follow a stepwise model. It is thus very important to check that all allelle sizes at each locus is compatible with a stepwise model of mutation with a fixed motive lentgh (i.e. all allele size differences have to be a multiple of its motive size). If some allele sizes are incompatible with the motive length, then the user should check the chromatograms to verify if the size of the allele was correctly inferred, and possibly remove all occurences of this allele form the data set (i.e. set it to missing data `000`).

1.4 Output and file system

Here is a quick reference list of files read and written during `Migraine` usage. Some additional files are not described here as one should not edit them in normal use. The two main output files, as shown in the "Quick start" example, are the `results_`*n*`.txt` and `Rplots_`*n*`.eps` files. Other notable output files are

- `pointls_`*n*`.txt`: this is where likelihood values are written for all points, and is read by `R`;
- `pointls_`*n*`.old.txt`: A preexisting `pointls_`*n*`.txt` is saved under this name when a new one is created;
- `migraine_`*n*`.R`: written by `Migraine` and read by `R`, this file contains `R` code to be executed;
- `R_out_`*n*`.txt` file: `R` log file, which stores the verbose and sometimes obscure output that goes to the screen in an interactive `R` session;
- `nextpoints_`*n*`.txt` file: more on this one in Section 1.5;
- `nextpoints_`*n*`.old.txt` file: a pre-existing `nextpoints_`*n*`.txt` file is saved under this name just after being read by `Migraine`.
- `output_`*n*`.txt` file: this contains roughly the same information as the `results_`*n*`.txt` file, but in a more computer-friendly, and less user-friendly, form.

The only input files are the data file, and the settings file (default name: `migraine.txt`).

1.5 Iterative analyses

To continue on either the `LinearIBD` or the `OnePpoVarSize` minimal example, edit the `migraine.txt` file with a text editor, uncomment the line

`WriteSequence=ReadPoints,Append`

save the settings file and rerun `Migraine` as in the previous example (or alternatively, run the command line `migraine WriteSequence=ReadPoints,Append`). `Migraine` will read the parameter points in the file `nextpoints_1.txt`, estimate their likelihood, add the results to the previous `pointls_1.txt`, and `R` will again be called for all further steps. If this fails, then perhaps you need to set the `R` path as explained in footnote 1 of the long version of this documentation.

This example shows that it is easy to perform iterative analyses. Here `nextpoints_1.txt` contained points with a predicted high likelihood. A new `nextpoints_1.txt` is written at each iteration, so that starting with a few points in a wide parameter space, one can gradually narrow the exploration of the parameter space to better explore the high-likelihood region (see Section 7.1.4 for more details about those iterative analyses and the associated syntaxe to be used in the settings file).

2 Likelihood estimation using `Migraine`: background

This Section describes some methods used by `Migraine` in a non-technical way so that one can quickly use the program efficiently.

In `Migraine` the likelihood is estimated by simulation. Further, a likelihood surface is estimated (or "predicted") from the estimated points. The quality of these estimations will depend on various numerical settings, *briefly* introduced in this Section and more systematically described in later ones. Detailed descriptions of algorithms and of their properties, as well as of general statistical background, are beyond the scope of this documentation. However, we first recall some basic properties of likelihood ratio-based intervals and of the slightly less familiar profile-based intervals, which are the main basis for inference in `Migraine`. For a sound introduction to likelihood methods in general, see Cox & Hinkley (1974) or Cox (2006). For importance sampling algorithms used by `Migraine`, see de Iorio & Griffiths (2004a,b).

2.1 Confidence regions based on (profile) likelihood ratios

Confidence intervals/regions can be constructed from the likelihood ratio: a p-parameter point θ is included in the confidence interval if twice the logarithm of the ratio $L(\theta)/L(\hat{\theta})$, where $\hat{\theta}$ is the maximum-likelihood estimate, is above a given bound. This bound is given by the chi-square distribution with p degrees of freedom.

If the dimension of the parameter space is > 2, it is difficult to represent the confidence regions. Further, interest may be a in a composite parameter such as neighborhood (Nb) in the `IBD` models or for the ratio of present to past population size under the `OnePopVarSize` model, as well as for all other population size ratios under the `OnePopFounderFlush` model. For these time-inhomogeneous models, scaling the time by the mutation rate (e.g. $Dg * mu = D_{\text{in generations}} * \mu$) instead of scaling by population size (i.e. $D = D_{\text{in generations}}/2N_{\text{current}}$) may also be interesting, and `Migraine` now computes their point estimates and 1D profiles by default as extra-parameters (2Dprofiles and condifence intervals are not computed by default but using the keywords `oneDimCI` and `2Dprofiles`, see p. 54 and 56). However, the user can also choose to consider them as canonical parameters of the model using the keyword `TimeScale=MutationRate` (see p.43).

In these cases, profile confidence intervals/regions may be computed (Cox & Hinkley, 1974, p. 322; Cox, 2006). For example, the profile likelihood for Nb is the maximum value of the likelihood over all values of $2Nm, g$ which yield a given Nb. More generally, the profile likelihood for some parameter value(s) $\boldsymbol{\psi}$ is the maximum value of the likelihood consistent with the given ψ, i.e. the likelihood maximized over the parameters that are not part of $\boldsymbol{\psi}$ (the other parameters are thus *not* fixed to their maximum-likelihood estimates). p-values of the profile likelihood ratio tests are computed by the chi-square method, in the same way as generic likelihood confidence intervals. The number of degrees of freedom (df) is the dimension of $\boldsymbol{\psi}$. For example, a two-dimensional confidence region for $(2N\mu, \text{Nb})$ is deduced by comparing the profile likelihood ratio (actually, twice its logarithm, i.e. $2\{\ln[L(\hat{\theta})] - \ln[L(\theta)]\}$) to the χ^2 distribution with 2 df, while a confidence interval for Nb is deduced by comparing the same value to the χ^2 distribution with 1 df. A practical downside is that the profile likelihood computations may be slow as they require many maximizations steps. For three parameters, computing all one- and two-dimensional profiles may take a few minutes to a few hours, when the number of grid values for each parameter varies from 10 to 25 (as controlled by e.g. `GridSteps=10`). With four parameters, the one-dimensional profiles alone may take several hours. When profile likelihood computations are slow, the user can choose specific parameters, or pairs of parameters, for which 1D and/or 2D profiles will be computed using the `1Dprofiles` and `2Dprofiles` settings.

Since version 0.5.2, profile computations can be parallelize in R using the keyword `CoreNbrForR` (see **??**).

2.2 Accuracy of estimation of likelihood in each parameter point

Accuracy of estimation of likelihood in each parameter point depends on the number of replications of the estimation algorithm[2]. This number is given by the `NrunsPerPoint` setting. For all time-homogeneous models currently implemented (e.g. without past demographic changes, `IBD`, `OnePop`, `2Pop`), a remarkably low value of 5 often appears enough to get a good estimation of the likelihood surface, and more than 100 does not appear useful (Rousset & Leblois, 2007, 2012). A value around 30 is generally a good choice. It is nevertheless advised to increase those values to get final estimates. For time-inhomogeneous models (i.e. `OnePopVarSize`, `OnePopFounderFlush`), the importance sampling distribution can be much less efficient. For those models, preliminary analyses can be run with a value of 200 and a value of 2,000 iterations will give reliable results for most demographic situations. However values of 20,000, or even 200,000 are sometimes necessary for strong disequilibrium situations, see Section 8.2. On the other hand, values of 200-500 are generally sufficient for weak disequilibrium scenarios. For demographic situation with potential strong and recent past change in population sizes, it is advised to check whether increasing the number of replicates by a factor 5 or 10 changes the results of the analyses (point estimates and CI, as well as the diagnostic plot of the kriging showing the variance in the estimation of the likelihood, see such examples in Section 8.2).

Analyses with stepwise models (`SMM` and `GSM` to a lesser extent) and for the `ISM` also implies less efficient IS algorithms than for the `KAM/PIM`, and the number of runs per point should also be increased when using these mutation models (see section 8 for various examples).

2.3 Accuracy of likelihood surface prediction

To find the maximum likelihood estimates and confidence intervals, smoothing is used to predict the value of the likelihood in any point. The accuracy of the results will depend on the number and location of points sampled, and on the quality of the smoothing procedure.

2.3.1 Number and location of points

`Migraine` estimates likelihood in a given number of parameter points (controlled by the `PointNumber` setting), each point being a set of values for all

[2]the number of independent genealogies constructed by the importance sampling algorithm for strict likelihood, and the number of ancestral sequences for PAC-likelihood

canonical model parameters (i.e., for `IBD`: $2N\mu$, $2Nm$, g). `Migraine` has several options for exploring the parameter space, but in the first run it simply divides the range for each parameter in a given number of cells, and samples one point uniformly within each multi-parameter cell created in this way.[3] The initial parameter ranges are specified by the user through the `LowerBound` and `UpperBound` settings (as shown in the "first session" example). Uniform sampling may be performed on a log-transformed scale (or not) independently for each parameter, by using the `samplingScale` setting (detailed later).

As the damselfly example has shown, an iterative process allows one to refine the sampling of points. Interest is in the likelihood surface around its maximum, and points should be more densely sampled in this area. To that effect, the recommended settings are at least two iterations (e.g., `writeSequence=Over,Append` to perform the two iterations in a single run) and using points generated by `R` for the second iteration. The iterative procedure is illustrated in Rousset *et al.* (2018).

Improvements in the generation of the next points imply that it is better to perform more iterations with fewer points than in previous versions of `Migraine`. For example, 9 iterations with 200 points, instead of 3 iterations with 600 points as previously suggested, give much better results.

Under the `OnePopVarSize` model with a `GSM` (4 parameters), reliable results are generally obtained with 8 to 10 iterations and 200 points. For the `OnePopFounderFlush` model with a `GSM` with fixed parameter (also 4 parameters), we used 8 iterations with 400 points for the analysis of the Soay sheep data set presented in Rousset *et al.* (2018). Some analyses of real data sets under the `OnePopFounderFlush` model with a variable `pGSM` (5 parameters) required 10 to 15 iterations with 400 to 800 points to get enough points near the maximum of the likelihood surface (unpublished results). For these time-inhomogeneous models (i.e. with past changes in population size), more iterations (e.g. 12 to 16) may be required if the past change in population size is not clearly marked on the likelihood surfaces (i.e. likelihood surface not peaked) and/or if the initial parameter range specified by the user does not include the high likelihood zone of the parameter space.

Fewer total points are necessary in models where the likelihood is easier to estimate and/or with fewer parameters. In particular, in models of isolation

[3]This was confusingly called Latin hypercube sampling in de Iorio *et al.* (2005) and Rousset & Leblois (2007), although it may be seen as a practical approximation to "maximin" Latin hypercube sampling. Given that replicate estimates of likelihood are taken for some points, the present design may also be seen as a practical approximation to a roughly uniform distribution of tight clusters, which is useful for addressing the different needs of covariance estimation and prediction (Zimmerman, 2006).

by distance, 4 iterations of only 250 points each are generally sufficient to obtain results with biologically relevant accuracy. To ascertain the decimals and obtain nicer plots, it may be worth doubling the number of iterations.

2.3.2 Reliability of the smoothing (kriging) step

The method known as kriging is here used to compute a prediction of the unknown likelihood surface from the likelihood estimates in some parameter points . It is unwise to try to predict (extrapolate) the likelihood surface outside of the "kriged" parameter range, and in `Migraine` this is avoided in general: the maximum of the likelihood surface is sought only in the kriged range, the profile likelihood values too, and the plots clearly distinguish the kriged range. In `Migraine` the range is taken as the so-called convex hull of the kriged points (the minimal convex set containing all points, which one can visualize as a polyhedron with the most exterior points at its vertices).

Kriging depends on several smoothing parameters, which are estimated by `Migraine`. Ideally, the whole procedure used to infer the likelihood surface would work perfectly, so that users do not have to care about it. This is nearly so. In particular, the confidence intervals derived from the likelihood surface perform as expected in the simulation conditions described by Rousset & Leblois (2012); Leblois *et al.* (2014).

Nevertheless, users should check the screen output for two possible issues, namely (i) the messages warning that there may be too few points for some of the operations, in which case the obvious action is to compute more points by running an additional iteration; (ii) an estimated "smoothness" parameter lower than 4. The latter is really a problem only insofar it results in non-smooth likelihood surface, visible from the plots, and in poor prediction of the observed likelihood, visible from the diagnostic plot at the end of the `Rplots` file. In this case, see Section 2.6 of the long version of this document.

Estimation of the smoothing parameters is based on so-called generalized cross-validation (GCV). This may generate "`GCV...`" screen warnings during execution of `R` code, which can generally be ignored.

2.3.3 Parameter spaces and extrapolation

For `IBD` analyses, one can choose to infer the likelihood surface for the set of parameters ($2N\mu$,Nb$, g$), involving the neighborhood size Nb, rather than ($2N\mu, 2Nm, g$). However, if $2Nm$ and g were sampled uniformly, there may be wide gaps in the sampling of Nb values. Prediction of the likelihood surface for poorly sampled regions within the convex hull may be poor. Hence the prediction from kriging is good if sampling of points was roughly uniform on the scale used for kriging. For this reason, the parameter space for kriging (which is controlled by the `KrigSpace` setting) is by default the parameter space used to define uniform sampling of points (which is controlled by `SamplingSpace`).

This means that to make inferences about a composite parameter such as Nb, one should either sample uniformly Nb, then use it as a kriging variable, or should sample uniformly Nm, use it as a kriging variable, and use the resulting predictor to compute the Nb likelihood profile. The downside of the first option is that additional points must be sampled uniformly on a $2Nm$ scale if inferences are also made about $2Nm$. The downside of the second option is that it may lead to wrong extrapolation of the likelihood surface.

As noted above, profile likelihood methods provide a way of making inferences about composite parameters, such as the neighborhood size Nb viewed as a function of $(2Nm, g)$ dispersal parameters. But we also noted that prediction of likelihood values are safely made only within a set of points (convex hull) defined in the parameter space used for kriging, and this raises difficulties in computing profile maxima for parameters that are not part of the definition of this space. For example, the convex hull of points in $(2N\mu, 2Nm, g)$ coordinates is not the same as that of the same points in $(2N\mu, \mathrm{Nb}, g)$ coordinates. As a result, the maximum likelihood within the $(2N\mu, 2Nm, g)$ hull is not necessarily the same as within the $(2N\mu, \mathrm{Nb}, g)$ hull. On the other hand, only a convex hull in given coordinates [say $(2N\mu, \mathrm{Nb}, g)$] is a convenient set to explore in order to compute the profile for one of the coordinates, such as Nb. Therefore, when the tested parameter is not in the kriging space [here, Nb when kriging is in $(2Nm, g)$ space], a convex envelope is recomputed in a composite space $(2N\mu, \mathrm{Nb}, g)$ and all profile likelihoods are computed within this composite hull. But the resulting maximizing coordinates may not be in the original kriging space, and then the likelihood prediction is not reliable. In particular, when a high likelihood ratio is predicted in such points, this calls for extending the kriged range, hence the set of points for which likelihoods have been estimated, to such regions (the iterative procedure does this more or less automatically).

Profile plots if `Migraine` show both shading and contour line information. As in the "slice" surface plots, the contour lines also display extrapolation results while the shading represent only results within the convex hull used for kriging. In the profile plots, one can remove the dubious extrapolations by using the option `Plots=Cautious`.

2.4 Hints for good results

For all models, it is unwise to mix likelihood estimates with different `NRunsPerPoint`. It is even more unwise to mix PAC-likelihood and true likelihood estimates. Therefore, `writeSequence=...,Over,...` should be used to overwrite previous results whenever the `NRunsPerPoint` or the statistic are changed in a computation sequence, e.g. as in

```
writeSequence=Over,Over,Append
NRunsPerPoint=10,30
StatisticSequence=PAC,IS
```

by which points from the third iteration are added to those from the second one, all of them being true likelihoods estimates deduced from 30 replicates of the IS algorithm, not mixed with the PAC-likelihood estimates from 10 replicates computed in the first iteration.

Further recommendations for specific models are given in the Sections detailing each of them. In particular, minimal values of `NRunsPerPoint` for reliable inferences. A simple way to evaluate the impact of numerical settings on the accuracy of the final estimates is to run two independent analyses of the data, differing only by the value of the `ptSamplingSeed` setting (which controls which parameter points are randomly drawn for likelihood estimation). Independent runs should also give similar confidence regions. If you care about relative differences of a few percents, then consider adding one iteration and increasing `NRunsPerPoint` by a factor of 3–10 relative to the suggested values.

3 Mutation models

The following models are implemented:

3.1 K-alleles model

Currently the symmetric K-alleles mutation model is implemented for all demographic models.

3.2 Strict stepwise mutation model (`SMM`)

A strict stepwise mutation model (`SMM`), often used for microsatellite loci, is implemented for all demographic models except `IBD`. Under this mutation model, each allele is represented as the number of repeats or the size of the allele in base pairs and each mutation removes or adds a single repeat to the ancestral state.

3.3 Generalized stepwise mutation model (`GSM`)

A generalized stepwise mutation model (`GSM`) is also implemented for the `OnePop`, `OnePopVarSize` and `OnePopFounderFlush` demographic models. As in the SMM, each allele is represented as the number of repeats or the size of the allele in base pairs. Each mutation removes or adds X repeats to the ancestral state, where X follows a geometric distribution with parameter pGSM. Considering this mutation model adds a parameter in the analysis of any demographic model. `Migraine` considers a `GSM` with a relatively small number of alleles and reflective boundaries.

For both `SMM` and `GSM`, further settings `givenK` and `SMMstepSizes` allows one to control the number of allelic states and the motif lengths for the different loci. This is especially important for inferences under time-inhomogeneous models such as the `OnePopVarSize` model (see p.14).

3.4 Infinite Sites mutation model (`ISM`)

For sequence data `Migraine` assumes an infinite sites mutation model (`ISM`, Kimura, 1969). It is currently implemented for all demographic models except `IBD`. Under this model every mutation in the coalescent tree gives rise to a new segregating position (*i.e.* no back mutations). `Migraine` makes use of the importance sampling equations for `ISM` given by de Iorio & Griffiths (2004a,b) or the algorithm of Hobolth *et al.* (2008).

4 Demographic models

`Migraine` does not care whether data come from an haploid or diploid population (or even haplo-diploid) and therefore uses numbers of gene copies as a common currency for all cases. Hence, in the following, N should always be understood as is the number of gene copies per deme. For diploid populations, $2N\mu$ is thus an estimate of $4N_\mathrm{d}\mu$, and $2Nm$ an estimate of $4N_\mathrm{d}m$, where N_d is the number of diploid individuals per deme. In both parameters, $4N_\mathrm{d}$ can still be understood as $2N$, i.e. twice the number of gene copies at a locus in a deme. As any introduction to coalescent theory makes clear, the "2" here has nothing to do with diploidy but with the fact that the $2N\mu$ or $2Nm$ parameters describe relationship between pairs of gene lineages.

The following models are implemented:

4.1 Isolation by distance with geometric dispersal

In this model, `Migraine` returns estimates of three "canonical" parameters $2N\mu$, $2Nm$, and g which is a scale parameter of dispersal distance, as further detailed below. It also reports estimates of the neighborhood "size" ($2N\sigma^2$ or $2N\pi\sigma^2$ for linear and two-dimensional habitats, respectively). The keywords `linearIBD` and `planarIBD` (see `DemographicModel` setting) are used to perform distinct analyses in one and two dimensions.

The issue of spatial units for neighborhood size: Nb is the neighborhood "size" parameter of IBD models. One- and two-dimensional habitats differ in the scale of neighborhood: individuals$\times$(spatial unit) in one dimension, individuals in two dimensions (Rousset, 1997). Hence, for **linear** habitats, neighborhood size depends on the unit of spatial distance used. This unit must be provided to the program in some way, as further detailed in a later Box.

In one dimension, dispersal to signed distance $k \neq 0$ can be described as

$$\frac{m}{2}(1-g)g^{|k-1|}, \tag{1}$$

for dispersal probability m and dispersal scale g. As explained below, however, this needs to be corrected in order to ensure that the total dispersal rate $2Nm$ is as expected despite edge effects, and then it only matters that dispersal probability is proportional to g^k (the m factor per se plays no role in the algorithm). For two-dimensional dispersal, the dispersal probability is first constructed as the product of one-dimensional probabilities, and then again corrected.

$D\sigma^2$ can be represented as $Nm\sigma^2_{\text{cond}}$, where σ^2_{cond} is the mean-square dispersal distance given that dispersal occurs. σ^2_{cond}, is a function of $g$only (formulas are given in the long version of this documentation), and can be estimated in the same conditions as g can: see the Appendix of Rousset & Leblois (2012) for a discussion of the robustness of g vs. neighborhood size estimation. `Migraine` can use the σ^2_{cond} parametrization (see `samplingSpace=,,condS2` setting) but these should not be misinterpreted as estimates of mean-square dispersal distances (σ^2).

Corrections for edge effects are defined so that the user-declared dispersal rate $2Nm$ is the maximum immigration rate over the different demes on the lattice, and that the dispersal model characterized by $(2Nm, g = 1)$ is the island model with immigration rate $2Nm$ in all demes. See the long version of this documentation for more details on this procedure.

4.1.1 Hints for good results

Minimal values for reliable inference under the IBD model are:

```
writeSequence=Over,Append <= at least two iterations
PointNumber=1000 <= the default value, 512, may be enough, but...
NRunsPerPoint=10
```

4.2 Nearest-neighbor stepping stone dispersal

The stepping stone model (in two dimensions, the canonical four-neighbors model) is the limit case of the geometric dispersal model with $g = 0$. One can constrain the analysis to this model by specifying the same `0` lower and upper bounds to g through the `LowerBound` and `UpperBound` settings.

4.3 Island model

This is also a special case of the geometric dispersal model with $g = 1$, and can be enforced also by specifying the same `1` lower and upper bounds to g.

4.4 Panmictic population at equilibrium

Samples can be analyzed under a model of a single panmictic population at equilibrium (`OnePop`) with a single parameter $\theta = 2N\mu$ for haploid and diploid data with N the number of genes of the population. Data from a single population can also be analyzed under a model of several populations, simply by specifying empty data for the other populations, but in such cases, dispersal parameters should affect the likelihood.

4.4.1 Hints for good results

For the `OnePop` model, the following minimal values will almost always give reliable results because likelihood computations are very efficient and there is a single parameter to infer:

```
writeSequence=Over  <= a single iteration is sufficient
PointNumber=100
NRunsPerPoint=10 <= 1 run/point is sufficient for KAM
```

4.5 Panmictic population with variable size

Two models of a panmictic population with variable population size are also implemented in `Migraine` to infer past changes in population size, their strength, time of occurrence and possibly their duration.

First, the `OnePopVarSize` model consider a single past change in population size. The change starts at some time $T + D$ in the past and finishes at some more recent time T. Because time is counted backwards, the more recent time T is smaller than the starting time $T + D$. This model has three or four parameters (not counting additional mutational parameters, e.g. for the `GSM`) which are (1) the scaled current population size $\theta_{\text{cur}} = 2N_{\text{current}}\mu$; (2) the scaled time $T = T_{\text{in generations}}/2N_{\text{current}}$ at which the change in population size terminated; (3) the scaled duration $D = D_{\text{in generations}}/2N_{\text{current}}$ of the population size change. The current version of `Migraine` has only be tested with parameter T set to 0 in Leblois *et al.* (2014), i.e. the change continues until the time of sampling; (4) the ancestral scaled population size $\theta_{\text{anc}} = 2N_{\text{ancestral}}\mu$. Inference of the composite parameter Nratio, the ratio of population size (Nratio $= N_{\text{current}}/N_{\text{ancestral}}$), is also implemented in this model and may allow easier detection of past change in population size. Last, inference of the composite parameters $Tg * mu = T_{\text{in generations}} * \mu$ and $Dg * mu = D_{\text{in generations}} * \mu$, which are times scaled by the mutation rate instead of scaled by current population size, are also implemented and may allow better interpretation of the timing of the events (more details below, at the end of the subsection).

Two different changes in population size are implemented in `Migraine` for the `OnePopVarSize` model and can be selected using the setting `VarSizeFunction`: (i) a discrete change in population size occurring at T; (ii) a continuous exponential change occurring between $T + D$ and T.

Second, the `OnePopFounderFlush` model considers two past changes in population size. Going forward in time, the first change is discrete/sudden and is directly followed by a continuous change as described above for the `OnePopVarSize` model. More precisely, at some time $T + D$ in the past, the population size change suddenly from the ancestral scaled population size $\theta_{\text{anc}} = 2N_{\text{ancestral}}\mu$ to $\theta_{\text{founder}} = 2N_{\text{founder}}\mu$.That is the first sudden past change. Then, the second continuous change, during which the size of the population change from $\theta_{\text{founder}} = 2N_{\text{founder}}\mu$ to $\theta_{\text{cur}} = 2N_{\text{current}}\mu$, begins (thus at $T + D$ in the past) and lasts until a more recent time T. As for the `OnePopVarSize` model, because time is counted backwards, the more recent time T is smaller than the starting time $T + D$.

This model has four or five parameters (not counting additional mutational parameters, e.g. for the `GSM`) which are (1) the scaled current population size $\theta_{\text{cur}} = 2N_{\text{current}}\mu$; (2) the scaled time $T = T_{\text{in generations}}/2N_{\text{current}}$ at which the continuous change in population size terminated; (3) the scaled duration $D = D_{\text{in generations}}/2N_{\text{current}}$ of the continuous population size change. Note that, as for the `OnePopVarSize` model, we have only considered T being null, i.e. the last change continues until the time of sampling; (4) the founder scaled population size $\theta_{\text{founder}} = 2N_{\text{founder}}\mu$; and (5) the ancestral scaled population size $\theta_{\text{anc}} = 2N_{\text{ancestral}}\mu$.

Inference of three composite parameters are also implemented in this model and may allow easier detection of past changes in population sizes: (1) Nratio, the ratio of current population size over the ancestral one (Nratio $= N_{\text{current}}/N_{\text{ancestral}}$); (2) NactNfounder-ratio, the ratio of current over founder population sizes (NactNfounder-ratio $= N_{\text{current}}/N_{\text{founder}}$); (3) NfounderNanc-ratio, the ratio of founder over ancestral population sizes (NfounderNanc-ratio $= N_{\text{founder}}/N_{\text{ancestral}}$).

For easier interpretation of the timing of events and for comparison with other programs, inference of the composite parameters $Tg{*}mu = T_{\text{in generations}}*\mu$ instead of T, and equivalently $Dg * mu = D_{\text{in generations}} * \mu$ instead of D, is now implemented by defaults for point estimates and 1D profiles under both models. The user can also use the `TimeScale` keyword to set them as time parameters of the sampling space under both `OnePopVarSize` and `OnePopFounderFlush` models (see p.43 for the keyword `TimeScale`).

As noted above, the `OnePopVarSize` and `OnePopFounderFlush` models have only be tested with three parameters, by setting $T = 0$. It is thus recommended to set $T = 0$, unless you really want to infer the parameter T, in which case a simple simulation study may be necessary to evaluate `Migraine`'s performances in such situation.

4.5.1 Hints for good results

For the `OnePopVarSize` model, and even more for the `OnePopFounderFlush` model, it is more difficult to obtain fast, reliable results than in previous models because the importance sampling algorithm is much less efficient for time-inhomogeneous models, especially when the population size change is strong and recent. It is thus advised to proceed in two steps and to check consistency of the results over two different runs with different `ptSamplingSeed` and different `NrunsperPoint`. For very recent demographic change, it is advised to consider more iterations (and thus more points in total) because likelihood surfaces may not be clearly peaked, and may show cross-or funnel-like shapes. It is also advised to compute more points and to consider more iterations when using the `GSM` because it increases the number of parameters by one. The following values should give reliable results unless demographic is recent and/or strong (e.g. Nratio > 100, or < 0.01; and $T < 0.25$).

For `OnePopVarSize` with `SMM`:

```
writeSequence=Over,Append,Append,Append <= more iterations may give better results
PointNumber=500
NRunsPerPoint=2000 <= more  runs per points may be
 necessary when Nratio is > 100 or < 0.001
```

For `OnePopVarSize` with `GSM`:

```
writeSequence=Over,Append,Append,Append,Append,Append,Append,Append
PointNumber=500
NRunsPerPoint=2000 <= more  runs per points
 if Nratio is > 100 or < 0.001
```

For `OnePopFounderFlush` with `GSM`: A first run with a fixed pGSM value deduced from preliminary analyses under the `OnePopVarSize` and `GSM` models may be useful to restrict the explored parameter ranges. Then a run with the estimation of pGSM and the following settings may give good results:

```
writeSequence=Over,Append,Append,Append,Append,Append,Append,
Append,Append,Append,Append,Append,Append,Append
PointNumber=500
NRunsPerPoint=2000 <= more  runs per points
 if Nratio is > 100 or < 0.001
```

4.6 2 populations with migration

Migraine can also consider a model of 2 populations exchanging migrants (`2pop`). In this model, there are 2 sampled populations of size N_1 and N_2 exchanging migrants at rate m_{12} and m_{21} per generations respectively, where e.g. m_{12} is the probability that an individual from population 1 had a parent in population 2 (i.e. backward migration rates).

Considering $N = N_1 + N_2$, this model is defined in terms of 4 parameters $\theta = 2N\mu$, $Q_1 = N_1/N$, $M_1 = 2N_1m_{12}$, $M_2 = 2N_2m_{21}$, where population sizes are expressed as the number of gene copies per population, and migration rates are backward immigration rates. For diploid populations, the model of 2-populations is still valid if migration is gametic. This model has been tested in de Iorio *et al.* (2005) for a stepwise mutation model and to a lesser extent in unpublished results for the `KAM/PIM` mutation model.

Inference of four composite parameters are also implemented in this model and may allow easier interpretation of migration rates and easier test of asymetric migration patterns : (1) NMratio, the ratio of $M1$ over $M2$ allows to detect and test asymetry in scaled migration rates (i.e. number migrants), (NMratio $= M1/M2 = N_1 * m_{12}/N_2 * m_{21}$); (2) mratio, the ratio of m_{12} over m_{21} allows to detect and test asymetry in unscaled migration rates, (mratio $= m_{12}/m_{21} = M1/M2 * (1 - Q1)/Q1$); (3) `m1overmu`, the ratio of m_{12} over μ (`m1overmu`$= m_{12}/\mu = M1/\theta/Q_1$); and (4) `m2overmu`, the ratio of m_{21} over μ (`m2overmu`$= m_{21}/\mu = M2/\theta/(1 - Q_1)$).

For easier interpretation of the timing of events and for comparison with other programs, inference of the composite parameters $Tg{*}mu = T_{\text{in generations}}*\mu$ instead of T, and equivalently $Dg * mu = D_{\text{in generations}} * \mu$ instead of D, is now implemented by defaults for point estimates and 1D profiles under both models. The user can also use the `TimeScale` keyword to set them as time parameters of the sampling space under both `OnePopVarSize` and `OnePopFounderFlush` models (see p.43 for the keyword `TimeScale`).

4.6.1 Hints for good results

For the `2Pop` model with `SMM` mutations, the following values should give reliable results unless migration rates or population sizes are small:

```
writeSequence=Over,Append,Append,Append,Append <= a third iteration may give bett
PointNumber=400
NRunsPerPoint=100
```

5 Canonical order of parameters

This section summarizes the canonical order of parameters in each model, which is essential information for e.g. entering parameters ranges in the correct order.

Isolation by distance: `2Nmu 2Nm g`;

`2Pop` (PIM or SMM): `2Nmu N1/N 2N1m12 2N2m21`.

`OnePopVarSize pGSM 2Nmu T D 2Nancmu` or only `2Nmu T D 2Nancmu` if a simple stepwise mutation model is assumed;

`OnePopFounderFlush pGSM 2Nmu T D 2Nfoundermu 2Nancmu` or only `2Nmu T D 2Nfounderm` if a simple stepwise mutation model is assumed;

Note that this order only concerns canonical parameters of each model. Composite parameters (e.g. Nb, Nratio's, Dgmu, Tgmu, NMratio, mratio, `m1overmu`, `m2overmu`, etc) should either be (1) a substitute for a canonical parameter (e.g. Nb), and then be in the same place (see 7.2.6), or (2) an additionnal parameter (e.g. $Nratio$'s) and then only the order of the canonical parameters is important.

6 Data input

6.1 Input file format

6.1.1 Genepop

Input files should follow the Genepop format (as defined in the latest version of Genepop, Rousset, 2008; See the latest Genepop documentation). For example:

```
example of input file for Migraine
loc1
loc2
pop
, 0101 0102
pop
, 0101 0102
```

where each line represents the genotype of one individual at different loci, and groups of individuals ("samples" from different "populations") are separated by pop statements (see the Genepop documentation for further details).

6.1.2 NEXUS

Sequence data can also be analyzed by Migraine and should be specified in the NEXUS format (more details on this format can be found here) **and** a Genepop file (see below). Each NEXUS file contains information about a single locus: either all unique haplotypes (*i.e.* no duplicate sequences) or all sequences of each sequenced individuals. When conducting analyses with multiple DNA sequence loci, a separate NEXUS file is thus required for each locus (the format of the Nexus file name is detailled in section 7.2.1).

Below is an example of a NEXUS file generated by the IBDSim software. Note however that this example contains an *extra* sequence with the label Anc. As Migraine assumes an infinite sites mutation model, this extra sequence corresponds to the ancestral/reference sequence for the sample (i.e. the MRCA). If the Anc sequence has not been specified by the user then Migraine automatically constructs an ancestral sequence using the most frequent allele at each nucleotide position. Note also that the ancestral sequence is an extra sequence in the dataset and therefore needs to be taken into account by the `ntax` keyword which specifies the total number of sequences contained in the `NEXUS` file.

```
#NEXUS
begin data;
dimensions ntax=9 nchar=12;
format datatype=dna symbols="ACTG";
matrix
Anc         AGCTAGCTAGCT
001         AGGGAGCCACCT
002         AGCAAGATCGCT
003         AGGGAGCCACCC
004         AGCAAGATCGCA
005         AGAGAACCACCT
006         AGCAAGATCGGT
007         ACCAAGATCGCT
008         ATCGAGCTATCG
;
end;
```

It is important to note that `Migraine` also requires the genotype information (specified in a `Genepop` format) associated with the sequence data in the `NEXUS` file. This implies that the labels used in the `NEXUS` file need to correspond to those in the provided `Genepop` file (see also 7.2.1). However, when the number of haplotypes is large and the data is only available in the `NEXUS` format, it can be quite strenuous to manually create such a `Genepop` file. In such case, the user can use the C++ source code provided with the sources of Migraine, in the archive called `sourcesNexus2GP`, compile it (simply with `g++ -O3 -o Nexus2GP nexus.cpp Nexus2GP.cpp`) and run the binaries/executables which will automatically extract the sequence information `NEXUS` files and create the `Genepop` files with the genotype information.

6.2 Spatial information (isolation by distance)

6.2.1 Preferred method

The spatial coordinates of each sample can be given as a pair of coordinates in the name field of the last individual of the given sample. Thus

```
Another example of input file for Migraine
loc1
```

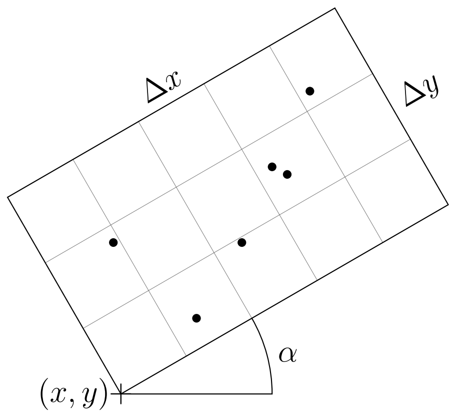


Figure 3: Meaning of `habitat` parameters
Six samples are distributed among fifteen bins, with two samples falling in the same bin.

```
loc2
pop
, 0101 0102
, 0101 0102
10 10, 0101 0102
pop
...
```

means that the first group is at position $(10, 10)$ in space.

However, this does not say the relative position of samples in the array of populations, which needs to be provided separately. Typically the position of spatially extreme samples are not the limits of the habitat, and thus one may need to specify the explicit shape of the habitat. This can be done using the `habitatPars` setting, as follows

```
habitatPars=297 15 500 300 30
geoBinNbr=5
```

The `habitatPars` arguments are, respectively, the x and y coordinates of a "lower left" corner of habitat, the dimensions Δx and Δy of a rectangular habitat, and a rotation angle α (in degrees) of this rectangle, as shown in Figure 3. The samples taken in this habitat are then binned in square bins covering the habitat. The largest dimension of the habitat is divided by the given number of bins (here, `geoBinNbr=5`), and the number of bins in the other dimension is deduced from this computation (hence, the total number of bins is not `GeoBinNbr`; you might prefer to use the keyword `AxialBinNbr`, with the same effect, to remember this). If one has data from a grid $(1,1)\dots(n_x,n_y)$ of positions and wishes to match this in the analysis, one should thus use

`habitatPars=0.5 0.5` $\Delta x = n_x$ $\Delta y = n_y$ `0`
`AxialBinNbr=`n_x

where the corner coordinates (0.5,0.5) implies that subsamples will be centered in the middle of each bin (and if you halve `AxialBinNbr`, bin limits will match every other bin limit of the original `AxialBinNbr` specification).

The same binning method should also be used for linear habitats when coordinates are given in the `Genepop` input file. In that case one would typically set Δy (lower or) equal to $\Delta x/$`AxialBinNbr` (but always > 0), so that there is only one y bin. If there are more y bins, the y bin value will be ignored, so that all samples are projected on the (rotated) x axis. In this way an imperfect linear habitat can be analyzed as linear. Whether a linear or two-dimensional model best applies to an elongated habitat depends on sampling design relative to habitat shape (Rousset, 1997). On the other hand `Migraine` will reject attempts to analyze an apparently linear habitat (a single bin in the y axis) as two-dimensional.

One can check how the data are binned by using

```
writeAdHocFiles=T
```

This will write files (typically named `re_dg`n for the nth locus) as as table where is column stands for a different allele, and each row gives the allele counts for each bin, with rows being ordered as (1,1), (1,2),..., (x-bin max,1),... ,(x-bin max,y-bin max).

6.2.2 Other methods (linear habitat only)

The habitat parameters can be deduced from settings alternative to `habitatPars`. See Section 6.2.2 of the long documentation.

6.3 The graphic output for the different models

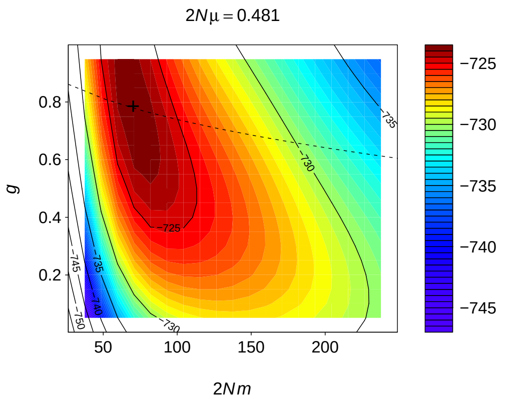
As described above, the `R` code produce various plots of the likelihood surfaces. The fast, default plots are grainy, as they represent grid of values computed for a limited number of values in each dimension, but this can be improved by increasing the value of the `gridSteps` setting, at the expense of a longer computation time.

With a minimal knowledge of `R` one can also locate the code controlling e.g. plot colors, and change it. We are interested to hear about the graphic needs of the users, though we cannot guarantee rapid and useful feedback on this matter.

6.3.1 Isolation by distance

Graphic output for this model were presented in Section 1.3.3.

"Slice plots" as shown in Fig. 1 (second plot) are computed for pairs of parameters in the kriging space. If the kriging space includes $2Nm$ and g, the $(2Nm, g)$ "slice" contour plot will include a dotted line showing $(2Nm, g)$ values with the same neighborhood size as the maximum likelihood estimate, as in the following plot from the damselfly example:

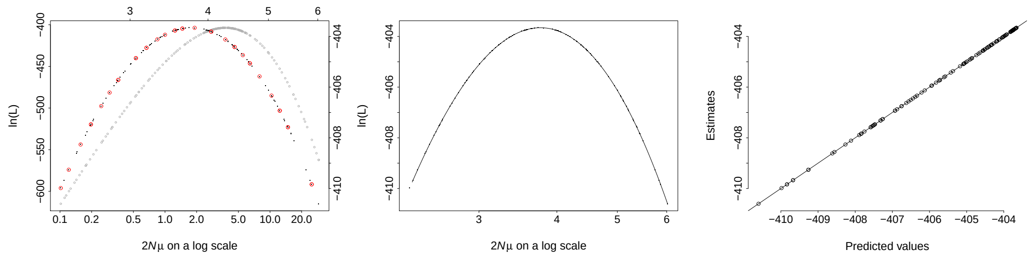


The issue of spatial units for neighborhood size, continued: As previously emphasized, for **linear** habitats, neighborhood size depends on the unit of spatial distance used. `Migraine`'s internal computations use bin width ("lattice unit") as the unit of distance, but the output (except some screen messages) is in terms of the "user unit", i.e. the unit of distance used for coordinates in the `Genepop` data file. The two differ by the number of user units per lattice unit; this multiplication factor is stored in the `GeoBinWidth` variable, which is reported is several output files, most notably in `results_`n`.txt`.

As shown in the session example, users can explicitly declare the `Nb` units shown in the plots as e.g. `GeoUnit=ind.m` (for individuals per meter, if coordinates were in meters).

6.3.2 Single panmictic population

For the single population model, three plots are produced:



First, a crude representation of the cloud of points, on two different scales as explained in section 1.3.3 for `IBD` and `OnePopVarSize` models, showing all points (in grey) on one scale, and on the other scale those selected for kriging (in black) and those selected for cross-validation (circled in red). Second, the likelihood curve for $2N\mu$ obtained by the kriging process. Third, the kriging diagnostic plot as described in section 1.3.3 representing the "observed vs. predicted" likelihood values for points selected for kriging.

6.3.3 Population with variable size: `OnePopVarSize` and `OnePopFounderFlush`

Graphic outputs for this model are very similar to those described in section 1.3.3 for the `IBD` and `OnePopVarSize` models, with the one-dimensional projections of the points are presented in two plots, one for all the points, the other for those selected for kriging and for cross-validation. There are 3 pairs of parameters under the `SMM` and 6 pairs under the `GSM`, hence 3 to 6 "slice" plots and 3 to 6 two-dimensional profile plots. Those figures are similar to the one plotted for the `2pop` model (Fig. 6). Some examples of `OnePopvarSize` outputs are shown if Fig. 4.

For the `OnepopFounderFlush` model, there are 6 pairs of parameters under the SMM and 10 pairs under the GSM, hence 6 to 10 "slice" plots and two-dimensional profile plots. Under this model, it is highly time consuming to compute all 1D and 2D profiles, and it is more appropriate to choose specific parameters, or pairs of parameters, for which 1D and/or 2D profiles will be computed using the `1Dprofiles` and `2Dprofiles` settings. The figures are similar to the one plotted for the `OnePopVarSize` model (Fig. 4). Some examples of OnePopFounderFlush outputs are shown in Fig. 5.

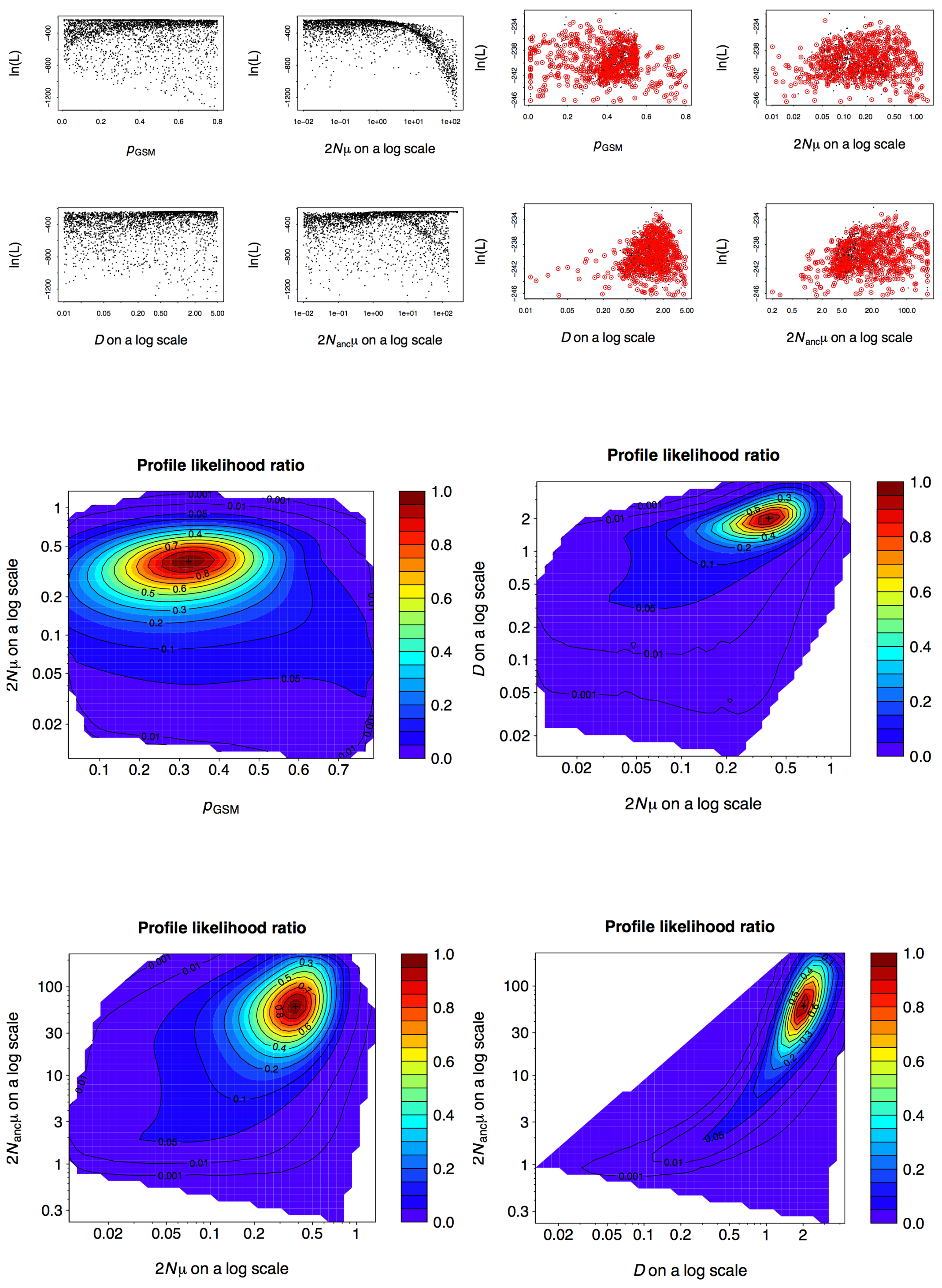


Figure 4: Examples of plots in the `OnePopVarSize` model with `GSM`. See section 1.3.3 and 6.3 for explanations of each type of plot.

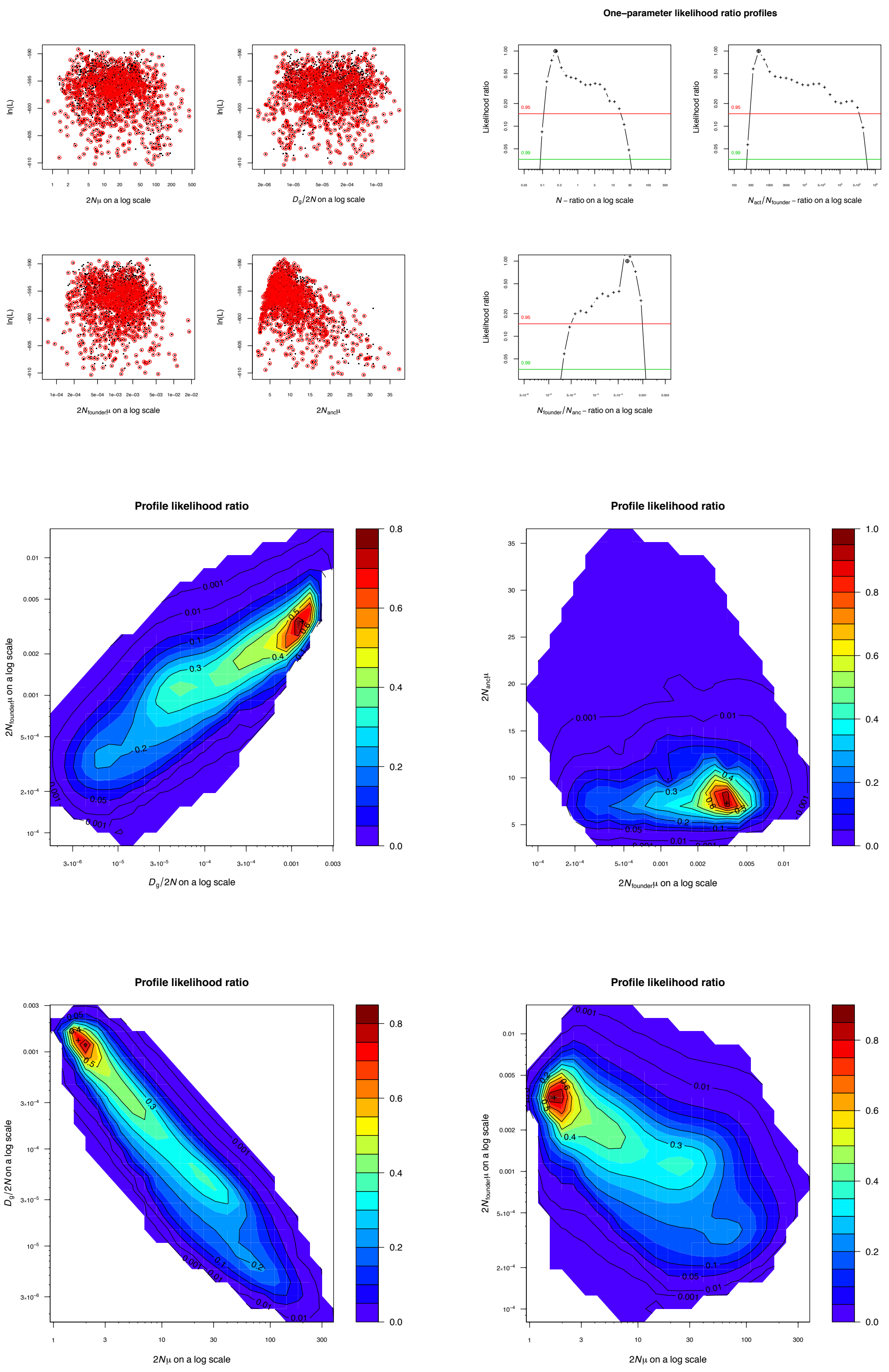


Figure 5: Examples of plots under the `OnePopFounderFlush` model with `GSM`. See section 1.3.3 and 6.3 for explanations of each type of plot.

6.3.4 2 populations with migration

Graphic output for this model are very similar to those described in section 1.3.3 for the `LinearIBD` and `OnePopVarSize` models, with the one-dimensional projections of the points are presented in two plots, one for all the points, the other for those selected for kriging and for cross-validation. There are 6 pairs of parameters, hence six "slice" plots and 6 two-dimensional profile plots. Some examples are shown if Fig. 6.

7 `Migraine` settings

Previous examples have shown several of the most important settings. This Section more systematically review the most essential settings. Many more are described in the long version of this documentation.

7.1 General features of settings

7.1.1 The settings file

The settings file allows one to control various settings of `Migraine`. It contains lines of the form *keyword=option*, where *option* can take various formats as described below. Any line not of the form `<Recognized keyword>=<option>` is ignored (empty `<option>` are allowed for some settings). Lines starting with `%`, `#` or `//` are ignored. Capitalization is not important except for path/file names under Linux.

The file is read at the beginning of execution. The file, or specific settings, may be missing, in which case case `Migraine` uses default values. These default values are set so that users can check that the program is running as expected before acquiring a full knowledge of the options; they do **not** represent suggested values for good performance.

The default name of the settings file is `migraine.txt`. You can change this through the command line:

```
migraine SettingsFile=mysettings.txt
```

will make `Migraine` read `mysettings.txt` rather than `migraine.txt`.

7.1.2 The command line

Options can also be given through the command line. However, in contrast to the settings file, one must avoid blank spaces as a separator within an option (e.g. `UpperBound=0.5 5 0.99` will not work). Commas will do, as in

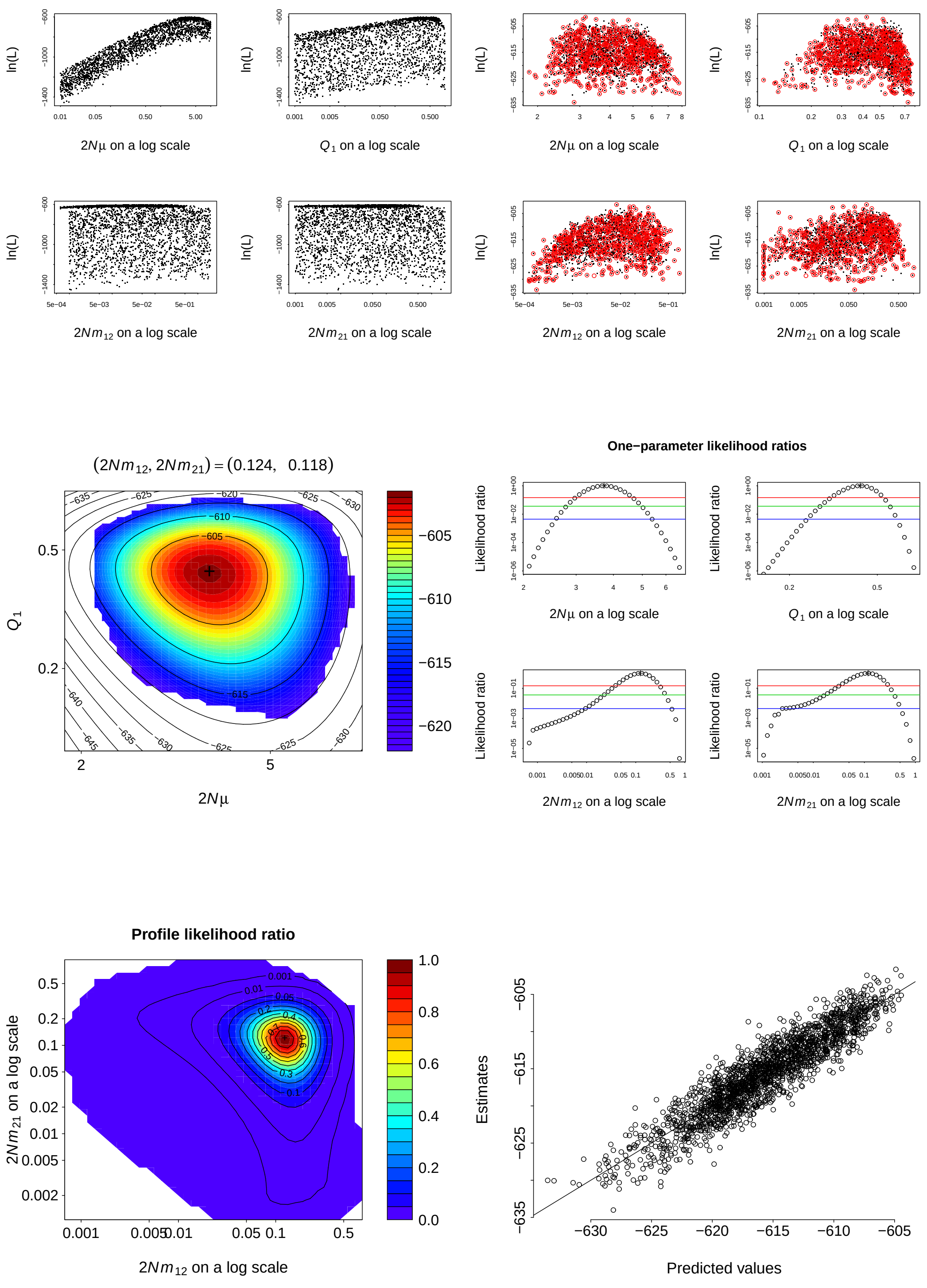


Figure 6: Examples of plots under the two-populations model. See section 1.3.3 and 6.3 for explanations of each type of plot.

```
migraine SettingsFile=mysettings.txt NrunsPerPoint=10 UpperBound=0.5,5,0.8
```

which makes `Migraine` read all settings in `mysettings.txt`, then override the `NrunsPerPoint` and `UpperBound` values given in this file.

7.1.3 Order of settings

The relative order of some settings matters. `PointNumber` will override previous values of `pointIndex`, `pointmin` and `pointmax`. `ParameterValue` overrides previous values of `LowerBound` and `UpperBound` by replacing both by the given `ParameterValue`, and each of them can also be overriden by new values. For safe use, follow a simple and logical sequence, as e.g. in the settings file used to produce the examples of Section 1.3.1 and 1.3.2.

7.1.4 The Iterations and Boolean syntaxes

As previously explained, `Migraine` makes it easy to perform iterative analyses where in each iteration, `R` can be called and the results of the `R` analysis can be used to fine-tune further computations in the next iteration, in particular by allowing a better exploration of the parameter space. Several settings can take different values over successive iterations, and thus their value is a list:

```
writeSequence=KrigOnly,Over,Over
StatisticSequence=PAC,PAC,IS
PointNumber=150,500
```

The first value is used in the first iteration, and so on... no trap here. The number of iterations is set by the `writeSequence` setting. Here, three iterations should be performed. If n iterations are called for by the `writeSequence` and there are less than n values (say m) in another list, the last value in that list is used in all iterations beyond the mth one. Thus in the above example, the second `PointNumber` and `writeSequence` values are used in the third iteration. Settings that follow this general rule are marked as iteration syntax in the following sections.

Note that since `Migraine` uses the `blackbox` R package (i.e. since v.0.5), it is now recommended to do more iterations with fewer points in each iterations (e.g. 4 to 15 iterations with 50 to 400 points each, depending on the dimension of the parameter space) than few iterations with lots of points. For example, under the `OnePopVarSize` model, 8 iterations with 200 points each should give better results than 2 iterations with 800 points each, as we previously recommended.

Booleans can be entered easily: they are False when the value given is `False`, `F`, `No`, or `N`; they are True when the value given is `True`, `T`, `Yes`, `Y`, or unspecified (as in "`writeAdHocFiles=`"). Settings that follow this pattern will be marked as Boolean syntax.

7.1.5 The locus vector syntax for analyses with multiple markers

`Migraine` can analyze different types of markers in a single analysis (e.g. tetra-nucleotide microsatellite loci following a SMM and di-nucleotides following a GSM). For such multi-marker analyses, several settings can take different values for the different markers, and thus their value is a list. Theses values can be set for each locus, or for groups of homogeneous markers defined by the `lociPerModel` keyword. For example, if one wants to analyze a set of 10 microsatellite markers, composed of 4 di- and 6 tetra-nucleotide unordered loci, by choosing a SMM for tetra- and a GSM for di-nucleotides, then the settings can be either

```
MutationModel=SMM,GSM,GSM,GSM,GSM,SMM,SMM,SMM,SMM,SMM
GivenK=200,40,40,40,40,200,200,200,200,200
SMMStepsizes=4,2,2,2,2,4,4,4,4,4,4
```

or, in a more compact form,

```
lociPerModel=1,4,5 <= defines groups of 1, 4 and 5 loci
MutationModel=SMM,GSM,SMM
GivenK=200,40,200
SMMStepsizes=4,2,4
```

If we assume the 7th and 8th loci to be sequence markers (*i.e.* under `ISM`), we can now write the example above as follows:

```
lociPerModel=1,4,1,2,2
MutationModel=SMM,GSM,SMM,ISM,SMM
GivenK=200,40,200,Auto,200
```

Note that `GivenK` has no meaning under the `ISM`, and should be set to `Auto`. `SMMStepsizes` also has no meaning under the `ISM`, and its value will not be considered.

Settings that follow this general rule are marked as locus vector syntax in the following sections.

7.2 Settings by theme

7.2.1 Data input

`GenepopFileName=mydata` (or `GenepopInputFile` or `GenepopFile`)

tells `Migraine` to read the `mydata` file in the `Genepop` format as shown in 6.1.1 .

`GenepopRootFileName`

for reading multiple `Genepop` files: see Section 8 of the long version of this documentation.

`NexusFileName=mydata` (or `NexusInputFile` or `NexusFile`)

tells `Migraine` to read the `mydata` file in the `NEXUS` format as shown in 6.1.2 . For anlayses with multiple DNA sequence loci, `Migraine` will automatically add '_locX' at the end of the `NexusFileName`, so that the name of the nexus file e.g. for the third sequence locus will be `NexusFileName`_loc3.nex.

`NexusRootFileName`

for reading multiple `NEXUS` files: see Section 8 of the long version of this documentation.

`Loci`

This setting serves to specify assumptions about the analyses of different loci. `Polymorphic` (the only option implemented so far) will thus consider monomorphic loci as uninformative and analyze only polymorphic loci. The default is to consider all loci as informative (monomorphic loci will thus drag down estimates of mutation rate, but other estimates should be affected as well).

`skipLoci`

This setting is used to exclude some loci from the analysis. It is a vector of intergers with each loci number that should not be considered for the inference. See also the keyword `KrigLoci` (section **??**) that can be used to select a subset of loci to be analyzed in R.

`NexusTaxonLabels`

The user needs to tell `Migraine` the type of the sequence labels contained in the `NEXUS` file. The following options `NexusTaxonLabels=Numerical` or `NexusTaxonLabels=other` are implemented.

`GenepopIndividualLabels`

The user also needs to specify the type of the individual labels contained in the `Genepop` file. The following options `GenepopIndividualLabels=NexusLabel` or `GenepopIndividualLabels=other` are implemented.

7.2.2 Spatial information

See Section 6.

7.2.3 Demographic models

`DemographicModel`

This setting specifies the demographic model to be considered for the analysis. The following options are currently implemented: (i) `linearIBD` for the linear habitat model and `planarIBD` for the two-dimensional habitat model; (ii) `OnePop` for the time-homogeneous panmictic population model; (iii) `OnePopVarSize` for the model of a panmictic population with variable population size; (iv) `OnePopFounderFlush` for the model of a panmictic population with variable population size; (v) `2pop` for the time-homogeneous 2-populations model. See section 4 for details on the different demographic models implemented in `Migraine`.

For the `OnePopVarSize` and `OnePopFounderFlush`models, some other settings are implemented:

`VarSizeFunction`

This setting specifies the type of population size variation considered under the `OnePopVarSize` model. The following options are currently implemented: (i) `Discrete` for a discrete change in population size occurring at T; (ii) `Exponential` for a continuous exponential change occurring between $T+D$ and T (Default). See section 4.5.

`TimeScale`

This setting specifies the scaling used for time parameters in non-equilibrium models (e.g. T and D for `OnePopVarSize` and `OnePopFounderFlush`). The following options are currently implemented:

(i) `PopSize` for inference of time scaled by population size (i.e. $T = T_{\text{in generations}} / 2N_{\text{current}}$ and $D = D_{\text{in generations}}/2N_{\text{current}}$); this is the default and recommended option;

(ii) `MutationRate` for inference of time scaled by mutation rate (i.e. $Tg * mu = \mu T_{\text{in generations}}$ and $Dg * mu = \mu D_{\text{in generations}}$). This is implemented for comparison with other programs (e.g. `IM, MIGRATE, DIYABC`,...) and is expected to give estimates with more variance than scaling by population sizes in most demographic situations Leblois *et al.* (2014).

Note that point estimates and 1D profiles of Tgmu and Dgmu are now computed by default (e.g. with `TimeScale=PopSize`) under the `OnePopVarSize` and `OnePopFounderFlush` models as extra- parameters. However, the user still needs to add those parameters in the keywords `oneDimCI` (or `1DCI`) and `2Dprofiles` settings to get confidence intervals and 2D profile plots for those parameters.

7.2.4 Mutation models

As previously described, a locus vector syntax is available to specify different mutation models for the different loci. For many loci, this is facilitated by the `lociPerModel` setting. Each mutation model is declared by `MutationModel` and further modified by `GivenK` and `SMMstepSizes` (depending on the model), as follows:

`MutationalModel` (or `MutationModel`) locus vector syntax

The following options are currently implemented:

`MutationModel=PIM` for a K-AM or PIM mutation model. The number of allelic states of the model is set as explained below with the setting `GivenK`.

`MutationModel=SMM` for a strict (i.e. single step) stepwise model (`SMM`, see section 3.2 for details), and

`MutationModel=GSM` for a generalized stepwise model (`GSM`, see section 3.3 for details), in which each mutation adds or removes X motifs of the microsatellite loci, with X following a geometric distribution with parameter pGSM.

`MutationModel=ISM` uses the infinite sites model (`ISM`, see also 3.4) for DNA sequence data type, in which each mutation adds a new segregating nucleotide position. Note that under this mutation model, `Migraine` can use the importance sampling equations given by de Iorio & Griffiths (2004a,b) (Default) or the algorithm of Hobolth *et al.* (2008) (with the setting `HobolthProposal=true`) .

In both the stepwise models, the name of each allele can either correspond to (1) its total length in bp or (2) the number of repeats of a microsatellite motif. For the latter case, one should also indicate the size of the repeated motif using the `SMMstepSizes` setting described below. This is especially important for inferences under time-inhomogeneous models such as the `OnePopVarSize` model (see p.14).

Reflecting **boundaries** are assumed for the GSM, and reflecting or circular boundaries are assumed for the SMM depending on the number of allelic states (see sections 3.2 and 3.3).

The **default number K of allelic states** is the observed number of alleles in the data for the `KAM`. For the `GSM`, $K = 40$ by default. However, this value may be too low for some loci. `Migraine` then outputs a warning message and automatically extend the number of allelic states to be equal to the number of observed allelic states + 10. For the `SMM`, the default K is 400, but can be overridden by the `SMMstepSizes` setting described below. K can be further controlled in all models by the setting `GivenK`:

`GivenK` (or `AllelicBounds`) locus vector syntax

`GivenK=`n allows one to override default values of number of possible allelic states. Different numbers of states may be specified for different loci. For example `GivenK=3,7` means that a 3-alleles model is assumed for the first locus and a 7-allele model is assumed for al further loci. This setting will be ignored if unfeasible (i.e. more observed alleles that given by `GivenK`).

For the GSM, although high values can be specified using `GivenK`, this is not recommended because computation times strongly increase with the number of allelic states.

For the `SMM`, whether circular or reflecting boundaries are considered depends on K. For given K, computations are faster in the circular case but **involve approximations which may be poor for large mutation rates and low** K.Hence it is highly recommended to set K to a large value; a finite circular `SMM` is considered in `Migraine` when $K \geq 200$, and reflecting boundaries are considered otherwise. 200 appears to give good results for most analyses, but high values of $2N\mu$ or very low migration rates necessitate larger values, e.g. 400 (the default) or even 600. Note that high K values may increase computation times.

Note that the `GivenK` keyword doesn't apply to sequence loci (*i.e.* `ISM`) hence one needs to specify the value `Auto` for such loci.

`SMMstepSizes` locus vector syntax

This setting controls the length of the repeated motif for each microsatellite locus. This setting is only necessary if the length in bp, and not the number of repeats, of each allele is given in the input file. By default, it is one for all loci, so that length in bp and in number of repeats are equivalent. The syntax is the following, for 6 microsatellite loci, 2 dinucleotide, 2 tri and 2 tetra:

SMMstepSizes =2,2,3,3 4 4 each value being separeted by a comma or a space. Note that SMMStepsizes also has no meaning under the ISM, and its value will not be considered for the ISM loci. This parameter and the fact that all alleles in the data set follow a stepwise model according to the motive length of the microsatellite locus is especially important for inferences under time-inhomogeneous models such as the OnePopVarSize model (see p.14).

lociPerModel locus vector syntax

This setting controls the different type of loci used for analyses with different type of markers. It is a vector of integers giving the number of loci for each type of marker (or each mutation model chosen for each marker), in the same order than in the Genepop file. The sum of the different values of lociPerModel is the total number of loci analyzed. For example, if one wants to analyze a set of 10 microsatellite markers, composed of 4 di- and 6 tetra-nucleotide unordered loci, by choosing a SMM for tetra- and a GSM for di-nucleotides, then the settings can be one of the two following options:

```
MutationModel=SMM,GSM,GSM,GSM,GSM,SMM,SMM,SMM,SMM,SMM
GivenK=200,40,40,40,40,200,200,200,200,200
SMMStepsizes=2,4,4,4,4,2,2,2,2,2,2
```

or more simply

```
lociPerModel=1,4,5
MutationModel=SMM,GSM,SMM
GivenK=200,40,200
SMMStepsizes=2,4,2
```

PromptForHyperSegSites

All sequence data to be analyzed by Migraine is expected to follow the assumptions of the Infinite Sites mutation model (ISM) which means that segregating sites are supposed to have only the ancestral and/or derived variants. In case of more than two polymorphisms at a given segregating site and the mention of this keyword (on a separate line in the settings file), Migraine interactively helps the user choose between eliminating the segregating site or the haplotypes contributing to the excess polymorphism.

ResolveForPerfectPhylogeny

Migraine expects that the provided sequence data in the NEXUS file satisfies the assumptions of the Infinite Sites mutation model (ISM). A standard way to verify for this is to perform a Four-Gamete Test (FGT, Hudson & Kaplan, 1985) which checks every possible pair of segregating positions and if it finds all four haplotypes (`00,01,10 and 11`) then it is evidence of a back mutation and the lack of a perfect phylogeny (Gusfield, 1991). The latter is a branching tree upon which mutations are assigned to specific branches and always exists for `ISM`-compatible data. This keyword can be assigned any of the following values :

`=manual` where the user resolves perfect phylogeny incompatibilities manually but with `Migraine`'s help

`=haps (or =haplotypes)` where `Migraine` automatically eliminates the haplotypes containing the nucleotide positions which contribute most to the failure of the FGT.

`=sites (or =nucleotides)` where `Migraine` automatically eliminates the the nucleotide positions containing the maximum number of FGT incompatibilities.

`=auto (or =automatic)` where `Migraine` automatically chooses to eliminate between the haplotypes and/or the nucleotide positions in order to eliminate all of the FGT incompatibilities.

7.2.5 Control of iterative computations

In each iteration, points are written in the `nextpoints_`n`.txt` file and read in the next `Migraine` iteration. Likelihoods are then computed for these points. The aim of the sampling procedure is to produce by successive iterations a well-centered likelihood ratio plot including all points with high likelihood (as determined by `NextboundsLevel`). It is advised not to alter the default behaviour. Nevertheless, the long version of this documentation describes how some settings can be used to control the sampling.

`NextBoundsLevel`

The confidence level used to define the bounds in the next step of the iterative algorithm (not to be confused with `CICoverage` used to determine confidence levels of intervals reported in output files). Default value is 0.001.

`WriteSequence` (Iterations syntax)

This setting determines whether previous contents of `pointls_`n`.txt` are appended or overwritten with the new series of points computed. In the first iteration, it also allows one to reuse the results of a previous run.

`WriteSequence=Append` will append the previously existing file, whether from a previous kriging iteration from the same run of `Migraine` or from a previous run. `WriteSequence=Over,Append` will overwrite any preexisting file in the first iteration, then append. The default is overwrite in all iterations (equivalent to `WriteSequence=Over`). Beyond `Over` and

`Append`, other options are `KrigOnly`, `WriteRKrig`, `ReadBoundsOnly`, and `WriteROnly`:

A final number n as in, e.g., `WriteSequence=`<...>`,Append,`n means: apply the latest stated operation (here `Append`) the given number n of times. Thus
`WriteSequence=Append,2,Over,Append,`2 would first accumulate the computations of the first two iterations, then overwrite them and accumulate the computation of the next three iterations. It obviously do not operate as the first value of `WriteSequence`.

`WriteSequence=`<...>`,noRcall` means: do not call `R` on the results of previous computations. It is meaningful only if preceded by a single `Over` or `Append` statement.

`WriteSequence=KrigOnly` means: Krig an existing file and continue. For example, `WriteSequence=KrigOnly,Append` will perform kriging of a preexisting "`pointls`" file using an existing `R` code, then append likelihood computations results to the "`pointls`" file.

`WriteSequence=WriteRKrig` means: write the `R` code and perform kriging of an existing data file. It operates only as the first value of `WriteSequence`.

`WriteSequence=ReadPoints` means: read points generated by a previous `R` run. It operates only as the first value of `WriteSequence`, typically followed by `,Append`.

`WriteSequence=WriteROnly` means: exits after writing the `R` code. It operates only as the first value of `WriteSequence`.

The last five options are meaningful only for the first iteration, and values of other iterative settings such as `StatisticSequence` are then meaningless for this first iteration. However, those other settings should also be given a value for the first iteration (so that one does not have to change them when one changes `WriteSequence`). For example, to achieve PAC-likelihood computation in the second iteration and IS in the third, one should set `StatisticSequence=IS,PAC,IS` or `StatisticSequence=PAC,PAC,IS`, even though the first IS/PAC term is ignored.

The code written in the `R` file depends depends on various settings read by the parent C++ process. Hence, *Do not analyze `Migraine` output obtained for some values of the settings with an `R` batch file written for other values* *of them*. Hence, only use the `R` file written by the same run as the likelihood computations analyzed, or produced from the same settings file with minimal alterations (settings that can be changed are `writeSequence`, `krigmax` and associated settings, graphic control settings, LRT and CI settings). It is easy to recreate the correct `R` code using the `writeRKrig` or `writeROnly` options.

7.2.6 Control of sampled points

`PointNumber` (Iterations syntax), `pointmin` and `pointmax`

These settings control the parameter points at which likelihood is computed. `PointNumber` is the total number of points considered in the parameter range. Its default value is 512 at the time of writing, but do not trust that. As for `NrunsPerPoint`, a sequence of values can be given, e.g. `PointNumber=1000,5000`, so that the ith value is used in the ith kriging iteration, and the last specified value is used for all further kriging iterations. See the different sections called "Hints for good results" to have more information about this setting.

Parameters points are sorted by increasing value (`order` function in `R`). Likelihood will be computed at `pointmax-pointmin` points out of them, starting with point `pointmin`+1. The default values are `pointmin`= 1 and `pointmax=PointNumber`, so that all `PointNumber` points are considered. When these default settings are changed, execution of `R` code is switched off[4] (it may be reset on by a later use of `writeSequence` in the settings).

`ptSamplingSeed`

This controls the random number generator for sampling of points. If you don't use this setting, `Migraine` should sample identical points in each execution. The seed should be an integer between 0 and 4294967295. The default value is 67144630.

`LowerBound` and `UpperBound`

This settings sets lower and upper bounds of the range of parameter values explored by the program (at the first iteration, at least). As `Migraine` can automatically expand the parameter range explore between different iterations (se section 7.2.5 "Control of iterative computations" for details), absolute minimal and maximal values for each parameter, i.e. that can never be exceeded during the whole iterative procedure, can be set using the `parMinima`

[4]This is useful as it prevents inference of likelihood surfaces from only a subset of all the points by each of a number of parallel runs of `Migraine` when the computation of many points is distributed among different processors.

and `parMaxima` settings describe below. See the description of each statistical model for further details.

`parMinima` and `parMaxima`

sets the absolute maximum and minimal parameter values to be explore during the iterative procedure.The argument is a vector of parameter names associated with numerical value, for any subset of the canonical parameters of the model. For example in a four-parameter `OnePopVarSize` model with a `GSM`,

```
parMinima=twoNmu=0.0001,D=0.0001,TwoNancmu=0.1
```

will set absolutes limits for the three demographic parameters of the model, so that values below those limits will never be considered at any iterative step by `Migraine`.

```
parMaxima=pGSM=0.7
```

will set an absolute upper limit for the `pGSM` parameter. All other parameters do not have any absolute upper limits in this case.

Note that parameter names are entered in an "ASCII-safe" style (`2Nmu` or `twoNmu` rather than 2Nμ); they can be omitted only if all canonical parameters are given in canonical order.

`samplingSpace`

Along with `samplingScale`, this setting helps define the parameter values that are sampled uniformly within the range given by `LowerBound` and `UpperBound`.

The statistical model defines some "canonical" parameters ($2N\mu$, $2Nm$ and g for the geometric dispersal model). The bounds define a range of values of the parameters to be explored, by default an (hyper)cube of values for these parameters. For all models but `IBD`, the default "canonical" sampling scale should be used and there is no clear reasons to use another space. But **for the IBD model**, this may not be most appropriate when only some combinations of parameters have a high likelihood. For example, the likelihood may be mainly function of the composite neighborhood (Nb) parameter, and in that case it is more interesting to investigate high Nm/low g values, and conversely, for fixed Nb values (both a high Nm and a high g value will result in an extremely large Nb value). Rather than sampling given ranges of Nm and g values, it is then better to sample given ranges of Nb and g values. The `samplingSpace` parameter specifies the meaning of the bounds. *Note that the parameters points (as they appear in* `pointls_`n`.txt` *and* `output_`n`.txt` *files, in particular) are still in terms of the canonical parameters of the model.*

There are two implemented deviations from the canonical set of parameters. The first is `samplingSpace=,Nb,` which specifies that the second parameter is the neighborhood size rather than $2Nm$. In the linear habitat model where Nb values depend on the spatial scale assumed by the user, all values of Nb input by the user and returned by the program are in the scale given by the user. The second is `samplingSpace=,,condS2` (or `samplingSpace=,,DispersedSigma2`). Together with `samplingScale=,,logscale`, this performs uniform sampling of $\ln(\sigma^2_{\text{cond}})$ rather than g (recall that σ^2_{cond} is the mean-squared dispersal distance of dispersed genes and is only a function of g). This can be generally useful when there is a plateau of high likelihood values for large values of the neighborhood size, as expected for samples simulated under high neighborhood values. See for example the analysis in Section 8.1. In contrast to the Nb scale, σ^2_{cond} is considered only a transformation of g and does not depend on the spatial scale. In other words, it is measured in lattice units. In these units it is at least 1 in linear habitats and 1/2 in two dimensional habitats.

`SamplingScale`

`SamplingScale=,logscale,` (say) indicates that the second variable is to be sampled uniformly on a logarithmic scale. Note that we assume that most users prefer not to input values on a log scale, so the meaning of the `Lower/UpperBound` values is unaffected by this setting (if one wants a maximum neighborhood value of 10^5, for example, this value, not its logarithm, should be declared through `Lower/UpperBound`).

Combining the `LogScale` option with either the `Nb` or `DispersedSigma2` options for `SamplingScale` is recommended when the signal of isolation by distance is weak; whichever of the alternative parameters is best to consider depends on the precision of Nm estimation (as can be assessed from the Nm confidence intervals). We suggest starting with `samplingSpace=,,DispersedSigma2` and `samplingScale=,,logscale` for populations from "continuous" habitats.

Using the `LogScale` option for parameters $\theta, (T,)D, \theta_{\text{founder}}$ and θ_{anc} is also recommended for the `OnePopVarSize` and `OnePopFounderFlush` models, at least for the first run. The reason is, because there is usually no a priori concerning past variations in population size, `Migraine` must explore very different parameter values to correctly infer all possible demographic scenarios (i.e. stable, contracting or expanding population).

Note that the kriging parameter space (including any log transformation) is by default the same as the sampling scale. It is possible to alter this behavior, using the `KrigSpace` and `KrigScale` settings described in Section 7.2.9 of the long version of this documentation, but this is risky.

7.2.7 Control of likelihood estimation

`NrunsPerPoint` (Iterations syntax)

The number of trees (for the IS algorithm) or sequences (for PAC-likelihood) considered in estimation of likelihood for each parameter point.[5] A sequence of values can be given, e.g. `NrunsPerPoint=10,50`, so that the ith value is used in the ith kriging iteration, and the last specified value is used for kriging iterations beyond the last one. See section 2.2 and the different sections called "Hints for good results" to have more information about this setting.

`StatisticSequence` (or simply `Statistic`) (Iterations syntax)

This setting controls the type of algorithm used to estimate likelihood or a heuristic approximation of it. Use `Statistic=IS` for strict likelihood analysis. With `Statistic=PAC`, PAC-likelihood computation is turned on, but it is feasible only for time-homogeneous models (i.e. `IBD`, `OnePop` and `2pop`). Note that whith `Statistic=IS`, exact analytical computations of the probability of the last pair of gene is used (see Rousset & Leblois, 2012; Leblois *et al.*, 2014), unless it is explicitly disabled by `Statistic=ISstrict`.

Further settings are specific to time-inhomogeneous models (i.e. `OnePop-VarSize` and `OnePopFounderFlush`), they are: `PACanc`, for using strict likelihood for the recent part of the coalescent simulation (i.e. from $T + D$ until present) and PAC-likelihood computations for the ancestral part of the coalescent simulation (which corresponds to an equilibrium population); and `PAC2id` for combining `PACanc` with analytical computation of the probability of the last pair of genes.

`UsePCL_SISR`

If `UseSISR=true` (default is `false`) the likelihood is inferred using the resampling procedure of Merle *et al.* (2017). Details about the algorithm and the different parameters used by this algorithm can be found in this publication. This setting can only be used with `StatisticSequence=IS` or `ISstrict`, not with `PAC`. When using the resampling algorithm, the following settings should also be defined:

> `SISR_Alpha`, is one of the tuning parameters , with `SISR_Beta` below, that are used to balance the effect of the information provided by the SIS weight and by the composite likelihood, respectively.

[5]default is 10 at the time of writing.

SISR_Beta is the second tuning parameter. These two settings takes values between 0 and 1.0. Section 4 (Results) in Merle *et al.* (2017) provides numerical examples showing the efficiency of the resampling distribution for a large range of values of the tuning parameters α and β. They shows that the influence of these tuning parameters is relatively weak (at least for $0.5 \leq$ SISR_Alpha ≤ 1 and $10^{-4} \leq$ SISR_Beta $\leq 10^{-2}$) and that values of SISR_Alpha=0.7 and SISR_Beta=0.01 are probably good for a range of scenarios. Larger SISR_Beta values may give better results for small sample size, and vice-versa.

SISR_EventsNbBetweenResampling sets the number of events between two successive evaluation of the ESS. At each of these steps, if the ESS decrease more than a given factor (see below SISR_ESSMinDecrease), then the algorithm resamples. Simulation tests suggest that it is better to resample as often as possible, that is with SISR_EventsNbBetweenResampling=1. With SISR_EventsNbBetweenResampling$\leq$0, the algorithm do not consider resampling.

SISR_CoaEventsOnly, If set to true, the resampling algorithm will resample (or more precisely compute and compare the ESS, see below) after a given number of coalescence events only, otherwise it will consider both coalescence and mutation events. Simulation tests suggest that resampling among histories with the same number of lineages (that is after a given number of coalescence events with SISR_CoaEventsOnly=true) is more efficient than considering mutation events too.

SISR_ESSMinDecrease sets the minimum value of the ratio between the ESS value computed at the previous resampling steps and the current ESS value (i.e. at the current step of the genealogy reconstructions) for which a resampling step will be carried out.

7.2.8 Options for likelihood ratio tests and one-dimensional confidence intervals

Likelihood ratio tests for given parameter values are computed using the testPoint setting. Precise one-dimensional confidence bounds may be computed using the oneDimCI setting.

Options for single-point tests

`testPoint`

The argument may be a vector of numerical values for any subset of canonical statistical model parameters. In the IBD model, the canonical $2Nm$ parameter can be replaced by an `Nb` value. One or more standard or profile LRT's can be computed. For example in a three-parameter IBD model,

```
        testPoint=twoNmu=0.08,twoNm=5,g=0.5
```

will perform a standard LRT with three degrees of freedom (df),

```
        testPoint=twoNm=5,g=0.5
```

will compute a single two-df profile LRT of the given $(2Nm, g)$ combination, and

```
        testPoint=twoNm=5
        testPoint=g=0.5
```

will compute two one-df profile LRTs; output order will be the same as input order. Note that parameter names are entered in an "ASCII-safe" style (`2Nmu` or `twoNmu` rather than 2Nμ); they can be omitted only if all canonical parameters are given in canonical order.

Options for confidence intervals

`oneDimCI` (or `1DCI`)

The argument is a vector of parameter names, or `All` to compute confidence intervals for all fitted and composite parameters. For each of these parameters, the bounds of the one-dimensional profile likelihood confidence interval for this parameter are computed, and reported as the last two elements of a line in `output_`n`.txt`. The coverage probability of the confidence intervals is controlled by the `CICoverageLevel`/`CIerrorLevel` settings. `NA` values are (still) reported after this computation for each bound which appears to be outside the explored parameter space. The relevant line of the output file is identified with a <*varname*>`_CI` string. All confidence intervals are also reported in the `Results_`n`.txt` file in a more readable form.

`CICoverageLevel`

As explained above, this sets the coverage probability of the CIs (the probability that they contain the parameter value). The default value is 0.95.

`CIerrorLevel`

gives exactly the same information in a complementary way: the probability that the interval does not contain the parameter value. The default value is 0.05.

Options for plots, including two-dimensional confidence regions

One- or two-dimensional (profile, if relevant) likelihood surfaces can be drawn. By default only a subset of the possible plots is produced, but more can be produced using the `Plots` setting.

`Plots`

controls the different plots produced by `Migraine`. In this way it also controls the computation of one- and two-dimensional likelihood profiles. Overall `Migraine` can produce the following types of plots: (1) Likelihood surface plots, presented either as "perspective" or as "contour" plots (as understood in `R`). When there is more than two parameters, the surface plots for a given pair of parameters assumes that all other parameters are fixed to their ML estimates. The default for all models is to plot surfaces for all pairs of fitted variables, only as contour plots; (2) One-dimensional likelihood profiles. The default is to ignore them for the `IBD` model, and to compute all 1D profiles for the others, except `OnePopFounderFlush` model(see below); (3) Two-dimensional likelihood profiles. The default for the `IBD` model is to plot only the $(2N\mu, 2Nm)$ and $(2N\mu, \text{Nb})$ confidence regions. The default for all other models, except `OnePopFounderFlush` model, is to plot likelihood profiles as contour plots for all parameter pairs.
For the `OnePopFounderFlush` model, .
The default behaviour for the plots can be modified by the following options that can be combined as in `Plots=All2DProfiles,BW,3D`:

`all1DProfiles` will additionally compute one-dimensional likelihood profiles for each fitted variable (and composite variables such as Nb, NMratio, mratio or Nratio's);

`all2DProfiles` will additionally compute two-dimensional confidence regions (as contour plots) for all pairs of fitted variables (and Nb, but not other composite variables such as NMratio, mratio or Nratio's).

`allProfiles` is equivalent to both `all1DProfiles` and `all2DProfiles`.

`noProfiles` tells `Migraine` not to compute any profile.

`B&WPlots` or `BW` will additionally provide black and white versions of the two-dimensional contour plots (of likelihood surfaces and of profiles).

`3DPlots` or `3D` will additionally provide perspective versions of the two-dimensional contour plots of likelihood surfaces.

Other settings controlling the format of figures are:

`1DProfiles`

takes a vector of parameters of the model (e.g. `1DProfiles=ParamX,ParamY,ParamZ`) and tells `Migraine` to compute and plot 1DProfiles for these parameters only. It can take composite parameters as input, such as Nb, NMratio, mratio, Nratio's, and the scale of representation of the composite parameters is then specified using the setting `extraScale` described below. This setting is especially usefull for models that have a high numbner of parameters, leading to high computation times of 1 and 2D profiles, such as the `OnePopVarSize` model.

Since version 0.5.2, profile computations can be parallelize in R using the keyword `CoreNbrForR` (see **??**).

`2DProfiles`

works exactely as `1DProfile` described above but takes a vector of parameter pairs (e.g. `2DProfiles=(ParamX,ParamY),(ParamX,ParamZ)`) instead of single parameters.

`extraScale`

By default the variables shown in plots are based on the kriging variables, but may involve an additional one (e.g., both $2Nm$ and neighborhood size are plotted in the `IBD` model, although kriging will consider only one of them). `extraScale` allows to specify the scale (log or not) for variables not considered in kriging, the most typical use being `extrascale=Nb=logscale` under `IBD`, `extrascale=NMratio=logscale,mratio=logscale` for `2pop`, `extrascale=Nratio=logscal` for `OnePopVarSize` , or `Extrascale=Nratio=logscale,` `NactNfounderratio=logscale,NfounderNancratio=logscale` for `OnePopFounderFlush`. Otherwise, `Migraine` may use the scale it considers best.

`graphicFormat`

Controls the graphic output format; default is `eps`. The other options are `postscript` and `pdf` as described in the `R` documentation (other file formats such as `png` are not considered because only one plot can be included per file for them). `postscript` produces EPS-compatible figures as `eps` does. However, the figure dimensions are different. With `eps`, the width and height can be controlled through the `graphicPars` setting.

`gridSteps` (Iterations syntax)

the number of grid values on each dimension of the grid plots (i.e. plots of the likelihood surface). The default value is 21. Larger values of `gridSteps` may help catching global maxima, and may result in nicer surface plots, but will take some more time.

`plotRange`

Used as e.g., `plotRange= Nb=50,2000; 2Nmu= 0.05 1`. This serves to explicitly control the range of values in grid plots (from 50 to 2000 for neighborhood, and from 0.05 to 1 for $2N\mu$, in this example). Note the syntax: specifications for different variables are separated by a semicolon, and the bounds for each variable are separated by a comma or a space (as usual, the space separator can be used in a settings file but not in the command line).

`graphicPars`

This allows the user to pass graphical arguments to `R`. Of course, adventurous users can play with the whole `R` code, at their own risk. Currently three types of arguments are handled. First, arguments can be passed to the `R` `par` function. These will affect all graphics (with exceptions for 3D plots produced using the `lattice` package). For example
`graphicPars=cex.axis=0.8,mgp=3,1,0`. See the `R` documentation for the meaning of the parameters. Second, the dimensions of the plots can be changed when the `graphicFormat` is `eps` (the default), using e.g.
`graphicPars=width=7,height=7` (dimensions are in inches per `R` convention). Third, the maximum number of ticks on the axes of plots can be controlled by `xmaxticks=`n and `ymaxticks=`n. Defaults are 9 and 10, respectively.

7.2.9 Control of kriging

`CovFnParams` and `fixedSmoothness`

`CovFnParams` is used to give scale parameters for each estimated parameter of the biological model. A scaled Euclidean distance between parameter points is given as argument to the MatÃ©rn correlation function. Likewise, `fixedSmoothness` is used to set the value of the smoothness parameter of the MatÃ©rn correlation function. The only sensible value is 4, the maximum used by `Migraine`. An estimated value < 4 may be indicative of problems with the input points. In old versions of Migraine, these settings could be used to provide the scale parameters obtained in a previous Kriging analysis, to bypass the slow cross-validation step. In more recent versions of `Migraine`, this is less useful, as the implementation of cross-validation is faster, and scale values estimated in one R run are used as initial value for scale estimation in the next R run, generally further speeding this step (the `nextpoints` file is used for communication between two successive iterations, so this works if no file manipulation has interfered with this communication). Nevertheless, these settings are useful for producing figures, as illustrated on p. 62.

`KrigSpace` (not for normal use)

If e.g. `KrigSpace=,Nb,`, kriging computations and subsequent plots are performed for neighborhood size rather than for $2Nm$. The syntax is the same as for `samplingSpace`, and the default value of `KrigSpace` is `samplingSpace`. Altering this default can give poor results.

`KrigScale` (not for normal use)

This setting (different from `KrigSpace`!) has two effects. First, it indicates that a given parameter should be log transformed for kriging. For example `KrigScale=,,LogScale` means that the values of the third parameter should be log transformed. The syntax is the same as for `samplingScale`, and the default value of `KrigScale` is `samplingScale`. Altering this default can give poor results. As for `samplingScale`, `KrigScale` can be used to achieve more "dome-shaped" surfaces. Second, beyond this aesthetic consideration, `KrigScale` affects the generation of the next parameter points by R. In particular these points will appears uniformly sampled on the kriging scale.

8 More examples

In this section, we show how to use `Migraine` efficiently.

8.1 Linear habitat: choosing a parameter space

Here we consider (unpublished, but real) data at 7 microsatellites in populations from a linear habitat. The samples roughly align along an habitat of length 1000m and a 25°angle relative to the horizontal (East-West) axis, hence the `habitatPars` value below. A first investigation with only 10 spatial bins and wide parameter bounds was conducted:

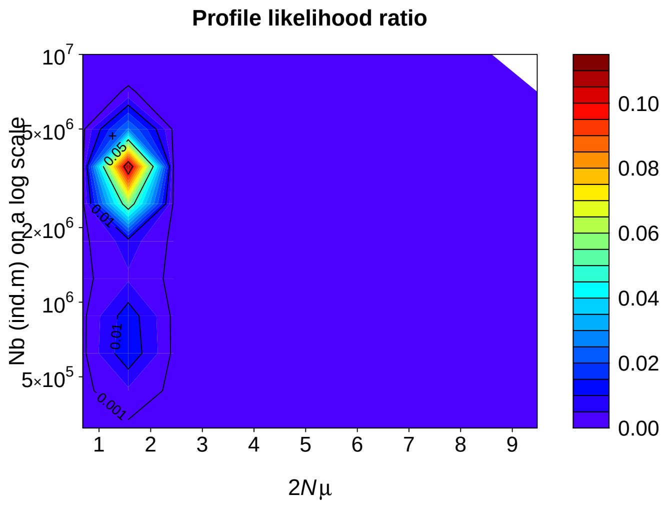
```
DemographicModel=LinearIBD
statistic=PAC
PointNumber=256
Nrunsperpoint=5
GeoDistanceBins=10
writeSequence=Over,Append
HabitatPars= 8600 -100 1000 1 25
GenepopFileName=unpublished.txt
BoundsDefs=,Nb,
samplingScale=,logscale,
Lowerbound=0.1,250000,0.01
```

```
UpperBound=10,12500000,0.999
GridSteps=11
writeAdHocFiles=T
GeoUnit= ind.m
```

The `PointNumber` and `NRunPerPoint` values are definitely too low, but these settings nevertheless put one on the right track, not the least because two iterations are nevertheless performed. The results file, obtained after a few minutes, indeed contain the following warnings

```
(!) Few points in upper 12.11 [ln(L) units] range:
   only 82 points in this range.
(!) No computed point has a predicted likelihood above  the one-dimensional CI thresho
    (threshold was -1.921 which is the 0.05 chi-square threshold with 1 df);
    It is advised to compute more points in order to obtain good CIs.
```

The plots are predictably poor-looking, but the most interesting one is the $2N\mu$,Nb profile plot:

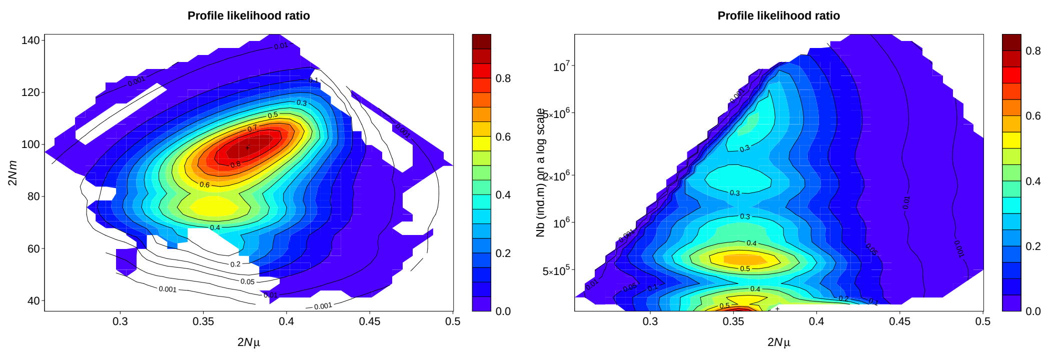


which shows where the interesting range of values is. In particular for $2N\mu$ it is roughly 1 to 2. In a more refined analysis we set

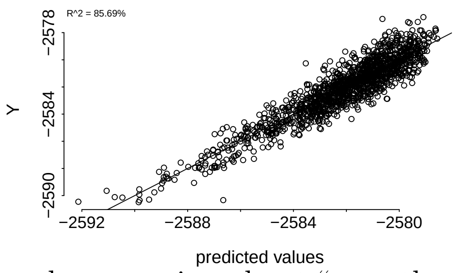
```
DemographicModel=LinearIBD
statistic=PAC
PointNumber=512
Nrunsperpoint=30
GeoDistanceBins=40
writeSequence=Over
writeSequence=writeRkrig,Append,3          <= four iterations
writeSequence=Readpoints,Append
```

```
HabitatPars= 8600 -100 1000 1 25
GenepopFileName=unpublished.txt
BoundsDefs=,Nb,
samplingScale=,logscale,
Lowerbound=0.25,250000,0.4
UpperBound=0.5,12500000,0.999
GridSteps=11,11,11,41          <= larger value in last iteration for nice plots
writeAdHocFiles=T
GeoUnit= ind.m
```

Note that interesting range $2N\mu$ is divided by 4 as the number of spatial bins is fourfold increased, because the N per bin correspondingly decreases. The neighborhood size could have been similarly adjusted by we rather choose to investigate a wide range of values. This computation takes a short night to complete, yielding the following profile plots:



The $2N\mu, 2Nm$ plot (left) is a nice example of modern art. Areas showing contour lines but no shading are the result of some extrapolation, as discussed in section 2.3.3, since kriging was performed on (log)Nb, not on $2Nm$. This feature can be modified as explained in the same section. Aesthetics aside, this plot is a reasonable output. The $2N\mu$,Nb plot is more questionable, exhibiting multiple maxima for Nb, which suggests that something went wrong. However, the diagnostic plot is a good-looking regression with Gaussian error:

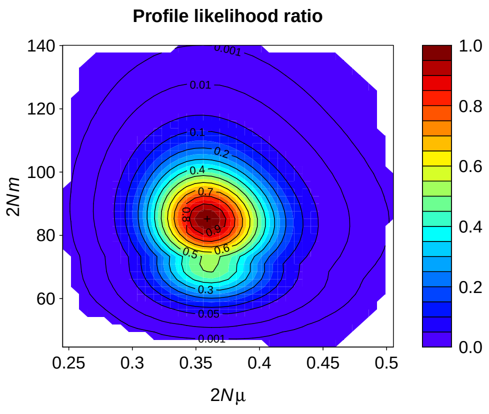


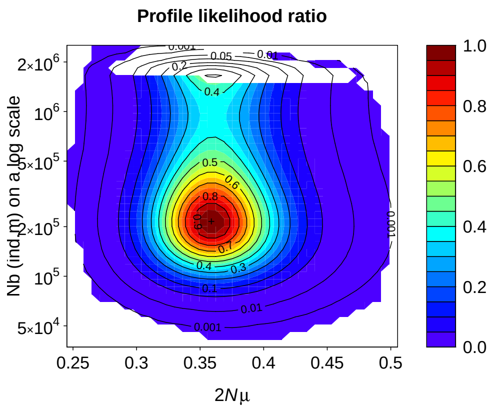
and no warning about “smoothness” was issued in the results file.

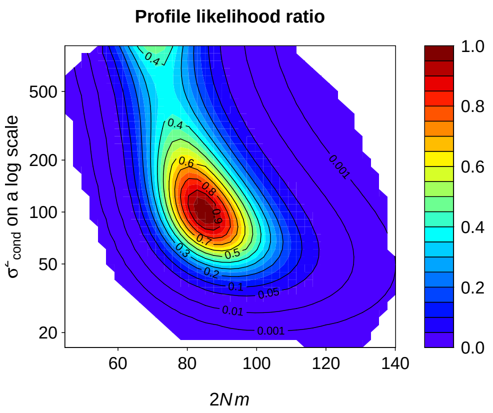
The plots could surely be improved by additional computations. However, the two profile plots show that $2Nm$ can be estimated with relatively good accuracy, while a wide range of Nb values have high likelihood. This suggests that $2Nm$, rather than Nb, should be used as a variable in the whole analysis, and then (log) `condS2` should be used instead of g to explore a wide range of large Nb values (see details of the `samplingSpace` keyword, p. 51):

```
...
BoundsDefs=,,condS2
samplingScale=,,logscale
Lowerbound=0.25,40,1
UpperBound=0.5,140,1000
...
```

The resulting profile plots are indeed nicer than (but broadly consistent with) the previous ones. Note that a $2N\mu$,Nb plot is still produced:







The plots suggest that there is a second maximum of the likelihood surface for high Nb or high σ^2_{cond} values, and this would be confirmed by better sampling of this parameter range (`UpperBound=0.5,140,100000`).

Despite what was shown in this example, it is worth fixing details of the figures (such as the horizontal position of the y-axis legend) before producing figures with high `GridSteps` values. In order to save some time in the recomputation of the figures, it is then useful to provide the covariance function parameter estimates obtained for a small `gridSteps` value (but an otherwise identical analysis), rather than recompute them each time. These estimates appear on screen and in the `R_out_0.txt` file output as

```
...
Cross-validation estimates of correlation function parameters:
    twoNmu      twoNm     condS2 smoothness
 0.3341275 75.7926204  4.1591363  4.0000000
...
```

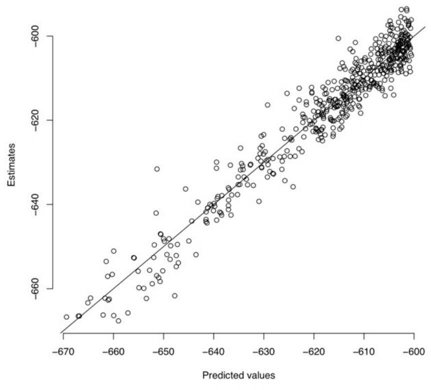
which translates into the following settings:

```
...
CovFnParams=0.3341275 75.7926204  4.1591363
FixedSmoothness=4
...
```

8.2 `OnePopVarSize` and `OnePopFounderFlush`: choosing the good number of runs per points

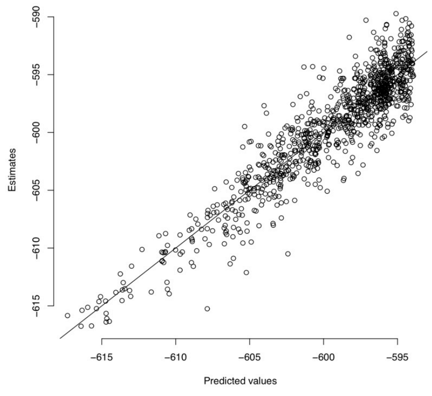
Running `Migraine` under the `OnePopVarSize` and `OnePopFounderFlush` models may not be always straightforward. The main reason is that IS algorithms are less efficient when strong and recent changes in population size occurred in the past, resulting in potentially biased inferences due to a high variance in the likelihood estimation at each parameter point. This problem is fully described and discussed in Leblois *et al.* (2014). We thus strongly advise any user (1) to read this paper and the current documentation; (2) to run the second example described at the beginning of this documentation (Soay shepp example, Section 1.3.2); and (3) then to read this section; before analyzing any real data set and interpreting the inference results.

Here, we continue on the Soay sheep example. For the first run, we considered only 20 runs (i.e. simulated coalescence trees) per points and only 2 iterations of 300 points each for a very quick analysis. With those settings, the run completes well within a few minutes, `Migraine` inferred a significant bottleneck as well as point estimates and relatively narrow CIs for all parameters. However, as we previously stated, difficulties to estimate the likelihood at each parameter point with enough precision can lead to



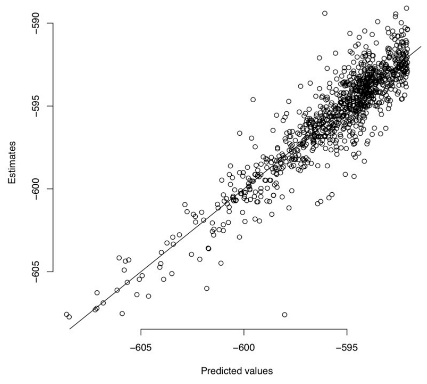
2Nmu : 0.28 [0.13 -- 0.51]
Dg/2N : 0.47 [0.24 -- 0.83]
2Nancmu : 6.1 [3.3 – 13]
Nratio : 0.046 [0.023 -- 0.096]
Dg*mu : 0.13 [0.041 -- 0.36]
3.40 5.76

(a) 20 runs per points



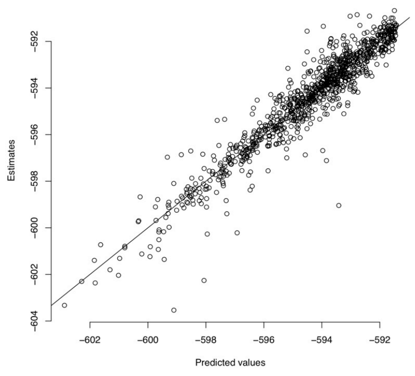
2Nmu : 0.19 [0.07 -- 0.39]
Dg/2N : 0.53 [0.36 -- 0.78]
2Nancmu : 7.4 [4.4 -- 14]
Nratio : 0.026 [0.0095 -- 0.06]
Dg*mu : 0.10 [0.034 -- 0.25]
1.85 29.2

(b) 200 runs per points



2Nmu : 0.14 [0.019 -- 0.34]
Dg/2N : 0.60 [0.39 -- 0.82]
2Nancmu : 8.4 [4.7 -- 16]
Nratio : 0.016 [0.0027 -- 0.044]
Dg*mu : 0.081 [0.012 -- 0.22]
1.09 28.7

(c) 2,000 runs per points



2Nmu : 0.11 [1.2e-05 -- 0.31]
Dg/2N : 0.63 [0.41 -- 1.5]
2Nancmu : 8.9 [5.8 -- 16]
Nratio : 0.012 [1.8e-05 -- 0.039]
Dg*mu : 0.068 [1.5e-05 -- 0.20]
0.53 62.4

(d) 20,000 runs per points

Figure 7: Point estimates, associated CIs and kriging diagnostic plots for different values of `nRunsPerPoints` from 20 to 20,000 for the analyses of the Soay sheep data set under the `OnePopVarSize` model presented a the second minimal worked example in Section 1.3.2.**Note the scale changes in log likelihood values between the four graphics.** We also report in red the last two numbers of the last line (i.e. named (final)) of the `output_`n`.txt` file, which are explained in Section **??**.

erroneous inference (Leblois *et al.*, 2014). It is thus important to check if the number of runs per point we chose (20) we choose was sufficient to get reliable results. Therefore the most important results to look at are (1) the diagnostic plot of the kriging (Fig. 7(a)): the diagnostic plot is a relatively good-looking regression with Gaussian error except for the higher likelihood values, for which the predicted values seems to flatten the estimated values. We can also note a relatively high variance in the estimation of the likelihood. (2) The variance of estimation is also indicated in the last line of the output ("final" stage), where the last two numbers refers to RMSpred the error of prediction and GOP the ratio of the theoretical error over the actual error of the fit (see In this first example with `nRunsPerPoints=20`, RMSpred is very large (i.e 3.4), but the kriging seems to correctly take this into account as the GOP is not very large and the profiles are smooth (see fig.2). Finaly, we can also note from the results_1.txt file (below), that there are probably not enough points at the top of the likelihood surface.

```
Migraine 0.5.5 (Built on Aug 16 2018 at 14:40:00)
blackbox, version 1.1.32 loaded
R code run on  Wed May 20 14:33:06 2020

Data file: Soay.txt
Settings file: migraine.txt

Demographic model: OnePopVarSize
Canonical parameters: pGSM 2Nmu Tg/2N Dg/2N 2Nancmu
* N stands for number of gene copies,
    i.e. 2N = 4 x [the number of diploid individuals] *

(!) Few points in upper 56.93 [ln(L) units] range:
   only 466 points in this range.
(!) Only 55 points have a predicted likelihood
     in the upper 1.921 [ln(L) units] range.
    (this threshold corresponds to the 0.05 chi-square threshold with 1 df);
    It is advised to compute more points in order to obtain good CIs.

*** Confidence intervals ***

95%-coverage confidence interval for 2Nmu : [ 0.13 -- 0.505 ]
95%-coverage confidence interval for Dg/2N : [ 0.243 -- 0.826 ]
95%-coverage confidence interval for 2Nancmu : [ 3.257 -- 12.75 ]
95%-coverage confidence interval for Nratio : [ 0.0225 -- 0.0956 ]
95%-coverage confidence interval for Dg*mu : [ 0.0406 -- 0.356 ]

*** Point estimates ***

     pGSM     2Nmu    Tg/2N    Dg/2N  2Nancmu
```

```
      0.5    0.282        0     0.47    6.062

      N ratio: 0.0464

      Dg*mu: 0.132

Normal ending.
```

At this stage, we know that (1) there are probably not enough points with high likelihoods for the smoothing procedure, and (2) the analysis shows high RMSpred values, indicative of a relatively large variance in the likelihood estimation, as shown in the kriging diagnostic plot. However, most importantly, we do not know if the number of runs per points was sufficient or not to give reliable inferences (i.e. if this high variance causes some bias in the analysis). The inferred parameters suggest that the past bottleneck may not have been very strong ($0.021 <$ Nratio < 0.091) and also not very recent ($0.48 < D < 0.84$), a situation in which the variance in the likelihood estimation should not be very large and thus not lead to erroneous inferences (Leblois *et al.*, 2014). We however choose to run three more analyses with more and more runs per points (and four iterations to get enough points in the top of the likelihood surface). All these changes should allows to get more points with high likelihoods and a smaller variance in the likelihood estimation.

To check the influence of the number of runs per points, we thus increased step by step the value of `nRunsPerPoints` from 20 to 20,000 as shown in Fig. 7. We can see that, as expected, increasing the number of runs per points strongly decrease the likelihood estimation variance (as shown by the kriging diagnostic plots and the RMSpred values). Unexpectedly, the GOP value increases with the the number of runs per points and reaches a quite high value of 62 for the last analysis, whereas the kriging diagnostic plot for this last analysis seems better than for the previous ones. Most importantly, we see that increasing the value of `nRunsPerPoints` also slightly shifts the estimates of all parameters : (1) $2Nmu$ decreases, as well as its CIs lower bound, and to a smaller extent its CIs higher bound; (2) $Dg/2N$ and $Dg{*}mu$ both increases, as well as their CIs lower bounds; and (3) $2N_{anc}mu$ and its CI lower bounds also increases. Those differences can reach up to a factor of 50 or 100% for some parameters between the first three analyses.

At this step, we do not know if we should consider more runs per points, and we could have considered a fifth analysis with 200,000 runs per points but it would have taken a very long time (e.g; many weeks on a desktop computer, few days on a computer grid). We choose not to run it because the last two analyses with 2,000 and 20,000 runs were highly concordant. The only noticeable difference between these last two analyses is a much lower CI bound for $2Nmu$, which reach extremely low values around 0.0001. Such low values are almost biologically unrealistic as a value of 10^{-5} corresponds to a population size of a single individual when considering a very small mutation rate for microsatellite markers of $2.5{\cdot}10^{-6}$). We thus do not believe that a fifth longer analyses would give quantitatively different and realistic estimates.

We thus showed that the analysis of this data set was sensitive to the number of runs per points considered. In other word, the inference results changed with the decrease in the likelihood estimation variance due to considering an increasing number of runs per points (Fig. 7). We could thus relatively clearly see the influence of increasing the value of `nRunsPerPoints`. Other data sets may even be more strongly affected the the variance in likelihood estimation under time-inhomogeneous models, so that analyses with few runs per points may lead to erroneous stable populations signals (personal observations).

We strongly believe that exploring different `nRunsPerPoints` values and comparing the results may be the only solution to check the reliability of a `Migraine` analysis under the `OnePopVarSize` and `OnePopFounderFlush` models. Comparing runs with `NRunsPerPoint`$= \{200; 2,000; 20,000\}$ should generally give a good idea of the accuracy of the inferences (but considering `NRunsPerPoint`$= 20,000$ may lead to high computation times and using a computer grid may be necessary).

All this argumentation is also valid for the `OnePopFounderFlush` model, which may even be harder to run because it is a more complex time-inhomogeneous model with an additional discrete change (i.e. the founder event) which will often corresponds to a very strong and quick change in population size, and thus a strong disequilibrium situation.

8.3 More examples

Check updates of this documentation for more examples!

9 Credits (code, grants, etc.)

`Migraine` uses R.J. Wagner's implementation of the Mersenne Twister random number generator. We thank the authors of the `R` packages used in this or previous versions of `Migraine`; in particular part of the kriging code has originally been derived from `fields`, C.J. Geyer offered feedback on `rcdd`, and D. Sterratt on `geometry`.

Initial collaboration with Maria de Iorio and Bob Griffiths has been supported by a CNRS/Région Languedoc-Roussillon Ph. D. grant to R. Leblois.

Over 2009–2013, this work has been supported by the ANR EMILE coordinated by J.-M. Cornuet and R. Vitalis. C. R. Beeravolu's postdoc was funded by the ANR IM-MODEL@Coral.Fish coordinated by Serge Planes and the INRA department SPE "Santé des Plantes et Environnement". C. Merle's PhD is jointly supported by the Cemeb and Numev "Labex'es". Computations have been performed on our institutes' grids (Institut des Sciences de l'Évolution in Montpellier, and Centre de Biologie pour la Gestion des Populations), and additionally on CINES (www.cines.fr) up to 2007, on sleeping computers of the University of Montpellier 2 for some time (thanks to V. Ranwez, K. Belkhir, G. Dugas, A. Weisseldinger, A. Dehne-Garcia, R. Dernat, and specially to J.-B. Ferdy), and on a grid from the Museum National d'Histoire Naturelle in Paris up to 2010.

10 Copyright

Bibliography

Cornuet, J. M. & Beaumont, M. A., 2007. A note on the accuracy of PAC-likelihood inference with microsatellite data. *Theor. Popul. Biol.* **71**: 12–19.

Cox, D. R., 2006. *Principles of statistical inference.* Cambridge Univ. Press, Cambridge, UK.

Cox, D. R. & Hinkley, D. V., 1974. *Theoretical statistics.* Chapman & Hall, London.

de Iorio, M. & Griffiths, R. C., 2004a. Importance sampling on coalescent histories. *Adv. appl. Prob.* **36**: 417–433.

de Iorio, M. & Griffiths, R. C., 2004b. Importance sampling on coalescent histories. II. Subdivided population models. *Adv. appl. Prob.* **36**: 434–454.

de Iorio, M., Griffiths, R. C., Leblois, R. & Rousset, F., 2005. Stepwise mutation likelihood computation by sequential importance sampling in subdivided population models. *Theor. Popul. Biol.* **68**: 41–53.

Gusfield, D., 1991. Efficient algorithms for inferring evolutionary trees. *Networks* **21**: 19–28.

Hobolth, A., Uyenoyama, M. K. & Wiuf, C., 2008. Importance Sampling for the Infinite Sites Model. *Statistical Applications in Genetics and Molecular Biology* **7**: 32.

Hudson, R. R. & Kaplan, N. L., 1985. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics* **111**: 147–164.

Kimura, M., 1969. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics* **61**: 893–903.

Leblois, R., Pudlo, P., Néron, J., Bertaux, F., Beeravolu, C. R., Vitalis, R. & Rousset, F., 2014. Maximum likelihood inference of population size contractions from microsatellite data. *Mol. Biol. Evol.* **31**: 2805–2823.

Li, N. & Stephens, M., 2003. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics* **165**: 2213–2233.

Liu, J. S., 2008. *Monte Carlo strategies in scientific computing.* Springer Science & Business Media.

Liu, J. S., Chen, R. & Logvinenko, T., 2001. A theoretical framework for sequential importance sampling with resampling. In: *Sequential Monte Carlo methods in practice* (A. Doucet, N. de Freitas & N. Gordon, eds.), Statistics for Engineering and Information Science, pp. 225–246. Springer New York.

Merle, C., Leblois, R., Rousset, F. & Pudlo, P., 2017. Resampling: an improvement of Importance Sampling in varying population size models. *Theoretical Population Biology* **114**: 70–87.

R Core Team, 2013. *R: a language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria.

Rousset, F., 1997. Genetic differentiation and estimation of gene flow from F-statistics under isolation by distance. *Genetics* **145**: 1219–1228.

Rousset, F., 2008. GENEPOP'007: a complete reimplementation of the GENEPOP software for Windows and Linux. *Mol. Ecol. Resources* **8**: 103–106.

Rousset, F., Beeravolu, C. R. & Leblois, R., 2018. Likelihood analysis of population genetic data under coalescent models: computational and inferential aspects. *Journal de la société Française de Statistiques* **159**: 142–166.

Rousset, F. & Leblois, R., 2007. Likelihood and approximate likelihood analyses of genetic structure in a linear habitat: performance and robustness to model mis-specification. *Mol. Biol. Evol.* **24**: 2730–2745.

Rousset, F. & Leblois, R., 2012. Likelihood-based inferences under isolation by distance: two-dimensional habitats and confidence intervals. *Molecular Biology and evolution* **29**: 957–973.

Watts, P. C., Rousset, F., Saccheri, I. J., Leblois, R., Kemp, S. J. & Thompson, D. J., 2007. Compatible genetic and ecological estimates of dispersal rates in insect (*Coenagrion mercuriale*: Odonata: Zygoptera) populations: analysis of 'neighbourhood size' using a more precise estimator. *Mol. Ecol.* **16**: 737–751.

Zimmerman, D. L., 2006. Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. *Environmetrics* **17**: 635–652.

Index